NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

HUMAN ROBOTIC SWARM INTERACTION USING AN ARTIFICIAL PHYSICS APPROACH

by

Brenton Campbell

December 2014

Thesis Co-Advisors: Timothy H. Chung
Richard M. Harkins

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12-19-2014	Master's Thesis 10-01-2013 to 12-02-2014

4. TITLE AND SUBTITLE

HUMAN ROBOTIC SWARM INTERACTION USING AN ARTIFICIAL PHYSICS APPROACH

5. FUNDING NUMBERS

6. AUTHOR(S)

Brenton Campbell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis explores the use of an artificial physics framework to provide centralized control of a collection of agents in close proximity to a human operator. Based on the spatial separation between agents, agents to way-point, and agents to operator, the artificial physics framework calculates virtual forces that are summed and translated into velocity commands. The virtual forces are modeled after real physical forces such as gravitational and Coulomb, forces but are not restricted to them, for example, the force magnitude may not be proportional to one divided by separation distance squared. These virtual forces allow the collection of agents, or the *swarm*, to autonomously find the operator, create a formation, and navigate way-points. The operator has high-level control of the agents via a hand held-controller. This framework is applicable to a scenario where an operator in the field needs to work with several autonomous vehicles but is unable to devote a high-level of focus to controlling agent behavior. We implemented an artificial physics framework in two simulation environments and in physical indoor experiments with a team of three unmanned aerial vehicles. The results from the physical experiments show that an artificial physics-based framework is an effective way to allow multiple agents to follow a human operator inside a small arena with only minimal operator input.

14. SUBJECT TERMS	15. NUMBER OF PAGES
multi-robot coordination, swarm, user interfaces, human-robot teams	73
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

HUMAN ROBOTIC SWARM INTERACTION USING AN ARTIFICIAL PHYSICS APPROACH

Brenton Campbell
Lieutenant, United States Navy
B.S., California Polytechnic State University, San Luis Obispo, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2014

Author: Brenton Campbell

Approved by: Timothy H. Chung
Thesis Co-Advisor

Richard M. Harkins
Thesis Co-Advisor

Andres Larraza
Chair, Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the use of an artificial physics framework to provide centralized control of a collection of agents in close proximity to a human operator. Based on the spatial separation between agents, agents to way-point, and agents to operator, the artificial physics framework calculates virtual forces that are summed and translated into velocity commands. The virtual forces are modeled after real physical forces such as gravitational and Coulomb, forces but are not restricted to them, for example, the force magnitude may not be proportional to one divided by separation distance squared. These virtual forces allow the collection of agents, or the *swarm*, to autonomously find the operator, create a formation, and navigate way-points. The operator has high-level control of the agents via a hand held-controller. This framework is applicable to a scenario where an operator in the field needs to work with several autonomous vehicles but is unable to devote a high-level of focus to controlling agent behavior. We implemented an artificial physics framework in two simulation environments and in physical indoor experiments with a team of three unmanned aerial vehicles. The results from the physical experiments show that an artificial physics-based framework is an effective way to allow multiple agents to follow a human operator inside a small arena with only minimal operator input.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

NPS	Naval Postgraduate School
ARSENL	Advanced Robotic Systems Engineering Laboratory
CAVR	Center for Autonomous Vehicle Research
BSD	Berkeley Software Distribution
DOD	Department of Defense
GUI	graphical user interface
Hz	hertz
IMU	inertial measurement unit
IP	internet protocol
m	meters
m/s	meters per second
mm/s	millimeters per second
PID	proportional-integral-derivative controller
ROS	Robot Operating System
sec	seconds
SDK	software development kit
UAV	unmanned aerial vehicle
GCS	ground command station
AP	artificial physics
PFM	potential field method

HSRI	human swarm robot interaction
RVIZ	Robot Three-Dimensional Visualization Tool
MEMs	microelectromechanical systems
3D	three-dimensional
2D	two-dimensional
COTS	commercial-off-the-shelf
GPS	global positioning system
FAA	Federal Aviation Administration
NAVAIR	Naval Air Systems Command
UDP	user datagram protocol
SSID	service set identification

Acknowledgments

I would like to thank co-advisor Dr. Tim Chung for his guidance and vision. Without his prodding and focus, this thesis would have never come together. I would also like to thank my other co-advisor, Richard Harkins, for helping me through the thesis process, providing equipment, and being my sounding board.

My fellow Advanced Robotic Systems Engineering Laboratory (ARSENL) members also contributed greatly to my work. I would especially like to thank Mike Clement, Brad Davis, Nicole Ramos, and Blake Wanier. We spent many hours together trying to get multiple AR.Drones to not crash into each other.

I'd like to acknowledge the unwavering support and commitment of my wife. We became parents while stationed in Monterey, and Sarah has done an amazing job at being a mom and a spouse to a sometimes eccentric naval officer.

Most importantly, I'd like to thank Jesus Christ. You are my Lord and Savior, and it is to you I owe everything.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

The role of unmanned aerial vehicles in warfare continues to increase as the cost and technical burdens associated with these platforms decrease. According to Gertler's 2012 U.S. Unmanned Aerial Systems report to Congress [1], the Department of Defense (DOD) unmanned aircraft inventory from 2002 to 2010 increased by more than a factor of 40. Current DOD unmanned aerial vehicle (UAV) programs of record are mainly focused on procuring highly equipped platforms adapted to support many different mission sets. This approach has led to serious cost concerns for programs that were once considered low-cost replacements for their manned counterparts [1].

Outside of the military, the last decade has seen a drastic rise in the number of commercially available unmanned systems to perform tasks ranging from underwater weld inspection to home cleaning. One very interesting civil development in this field is the rise of the UAV home hobbyist. Now, for as little as a few hundred dollars, someone with basic electronics and programming knowledge can adapt a "toy" UAV, such as the Parrot AR.Drone, to perform sophisticated tasks like flying way points, video surveillance, and object avoidance. Hobbyists and researchers are able to tap into vast amounts of open source code, algorithms, and applications that make customizing UAV behavior even easier. Through the internet, many disparate universities, research labs, and individuals can easily collaborate "crowd-source" solutions to complicated UAV problems.

A potential alternative to the Department of Defense's high-cost, high-capability UAV strategy is to focus on developing large numbers of low-cost, low-capability platforms that operate in theater simultaneously and cooperatively. These platforms would be able to self organize into a swarm, communicate, and navigate with a high degree of autonomy. Each individual within the swarm can operate based on a simple rule set. When these individuals interact with each other, complicated collective behaviors can arise. In nature, ant colonies are a prime example of this. Although each ant appears to be operating on its own, the

ant colony as a whole is highly organized [2]. In the study of swarm systems, the terms organization and emergence are used to describe the nature of the collective behavior. "Organization" refers to the increase in order or structure of the collective, and "emergence" is when new collective behaviors, patterns, or properties arise as a result of interactions between individuals [3]. If one or several platforms malfunctioned or were lost, the swarm would reorganize and carry on the mission. Swarm behavior could be controlled by a human operator or a small team of operators at a distant base station or while embedded among the UAV on mission.

1.2 Objectives

There is a resurgence of swarm behavior research due to advances in vehicle navigation, communications, and processing abilities. For individual platforms to function together as a cohesive swarm, a framework for sharing information must be established for platforms to act on this information in support of mission goals. This framework is ideally scalable to allow for coordination between few or many UAVs and compatible with many different types of unmanned vehicles.

Because it is unlikely, at least in the near future, that large swarms will be operating completely autonomously, it is important to determine the most effective way for human operators to control swarm behavior and receive information about the environment. One goal of this area of research is to balance vehicle autonomy with the amount of required operator input. The benefits of maximizing autonomy include:

- Reduced required number of operators
- Minimized operator fatigue
- Quicker response times to environmental stimuli

Some possible problems with increased autonomy are reduced fault tolerance, the inability to plan for all variables encountered in the environment, and inability to adapt or create new behavior patterns as the problem changes.

This thesis addresses the problem operator-UAV interaction in the field with many quadrotors. In addition to simulation data, experimental data was collected in a controlled physical environment. The scenario approximates a soldier on patrol in a hostile environment who

is attempting to detect and localize threats. One goal is to design and implement an integrated system so that the UAVs take off as a group and self-organize into a formation around the user. As the operator moves through the environment, the drones dynamically maintain proper distance from the operator and each other. During the evolution the drones will actively scan the area for threats using the onboard sensors. If a threat is detected, the swarm will alert the user via a hand-held device.

1.2.1 Simulation

Simulation is a popular method for testing frameworks and examining swarm behavior. This method allows the researcher to make small changes to algorithms and rule sets and immediately collect data on swarm behavior. More importantly, this allows the tester to bypass all the logistical details associated with actually operating robots in a physical environment such as charging batteries, establishing communications links, and positioning vehicles. Despite the advantages of simulation, testing in a physical environment is required to validate rule sets and observe how the vehicles react to variables not considered in the simulation. This thesis uses simulation to test the artificial physics framework prior to implementing the designed framework on physical robots.

1.2.2 Human Swarm Robot Interaction

Determining the most efficient way for an operator to control a swarm of UAVs is a subset of swarm UAV research that is still in its infancy. One must consider how much autonomy the vehicles should have versus how much user interaction is required. Minimizing the amount of required operator interaction reduces the number of necessary personnel and allows the operator to focus on other facets of the mission. This thesis attempts to advance the area of human swarm robot interaction (HSRI) by allowing the operator to control some aspects of swarm behavior using a hand-held controller while standing among the agents. Part of our study is to create an algorithm that allows several drones to find a human operator and create a formation around him or her with little or no input given by the operator, see Figure 1.1.

1.2.3 Questions

Some research questions explored in this thesis include:

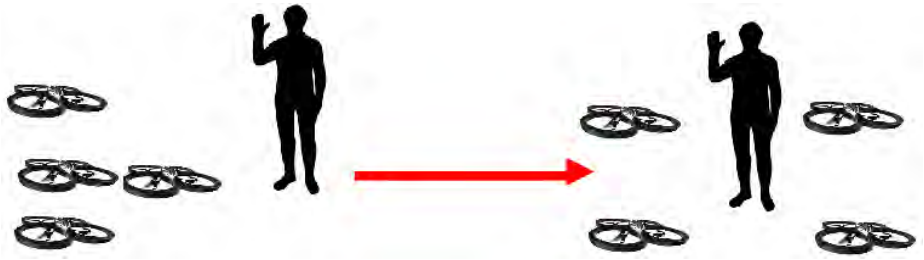


Figure 1.1: One goal of our research is to allow a group of UAVs to autonomosly create a formation around a human operator.

1. What rule sets must be employed to allow the swarm to form a formation, move, and avoid obstacles while not colliding with the operator or fellow vehicles?
2. What is the most efficient way for an operator to control swarm behavior in the field?
3. How will the swarm provide the user with information about the environment?

1.3 Benefits of the Study

This study provides insight into a proposed efficient method to balance UAV autonomy and operator interaction while in the field. The framework used in this experiment could be extended to UAVs operating in the outdoors using global positioning system data for position in lieu of a motion capture camera system. It also paves the way for creating algorithms to allow for more complicated behavioral control of the swarm such as directing an attack or having the swarm perform tactical reconnaissance on the path ahead. Coordinated swarms of conventional off-the-shelf quadrotor UAVs could be a powerful force multiplier in the expeditionary sensor grid while being much more cost-effective than existing high-end UAV programs of record. Although data taken from a single unit may be inconsequential, when aggregated with data from many other swarm members, a higher level of battlespace awareness is achievable. In summary, this study helps to understand the link between micro agent level behavior to macro behavior of the entire swarm while in the presence of human operators.

1.4 Thesis Organization

The next chapter reviews similar work on swarm robot control while highlighting some of the frameworks used to control multiple robots simultaneously. Chapter 2 also introduces physicomimetics or artificial physics (AP) on which the robot control framework developed

in this thesis is based. In Chapter 3, the various systems that make up the experimental design and their integration are discussed. Chapter 4 presents results and analysis from several experimental scenarios. Conclusions, recommendations, and future work are in Chapter 5. Bold font is used to represent vector notation, and hatted vectors denote unit vectors.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Related Work

The focus of this thesis is to explore the concept of an operator controlling many autonomous UAVs while physically located among them. These UAVs need to operate in a framework that is scalable, fault-tolerant, and inexpensive while demanding minimal direct control by the human operator. Over the past several decades, many different frameworks have been explored using simulation and physical experimentation.

Typically, robot swarm behavior is based on interactions between each member of the swarm and each member and its environment [4]. In swarm robotics literature, the words "microscopic" and "macroscopic" are used to differentiate agent level and swarm level behavior, respectively [5]–[7]. At the microscopic level, robots have limited knowledge through their sensors about the world around them. Nominally, at this level, changes in the perceived environment elicit very simple responses, e.g., if my left bump sensor detects a collision, turn right. At the macroscopic level, swarms can exhibit behaviors such as exploration, foraging, and flocking. Linking micro to macro behavior is the main task of swarm robotics.

2.1 Potential Field Method

One way to control robot behavior at the micro level is with a potential field method (PFM). Using artificial attractive and repulsive potentials to control unmanned vehicles in a swarm is not a new concept. This approach was initially applied to control a manipulator arm in a fixed environment with several obstacles [8]. The manipulator is repulsed by obstacles and attracted to targets. A simple algorithm calculates the potential energy function, $U(x,y)$, based on the separation distances to environmental boundaries, obstacles, and the target. Because force is proportional to the gradient of the potential energy function, Equation 2.1 allows us to calculate the force vector.

$$\mathbf{F}(\mathbf{x},\mathbf{y},\mathbf{z}) = -\nabla U(x,y,z) \tag{2.1}$$

The forces vector is converted to a desired change in velocity by Equation 2.2 where Δt is the amount time per update cycle and m is the virtual mass of the robot.

$$\Delta \mathbf{v} = \frac{\mathbf{F}(\mathbf{x}, \mathbf{y}, \mathbf{z})\Delta t}{m} \tag{2.2}$$

After this calculation, $\Delta \mathbf{v}$ is converted to a control signal to the robot's motors. This allows for basic path planning by choosing the path with the steepest gradient ultimately leading to a the position where potential energy is minimized. This simple, but elegant framework, now known as the PFM, is applied to many different problem sets because it is quick to implement and usually yields acceptable results [9]. Figure 2.1 depicts a typical potential field scheme for navigating a robot towards a target way point.

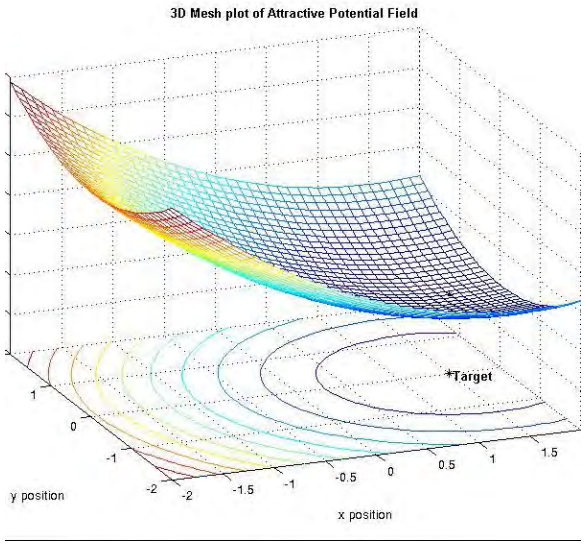


Figure 2.1: A potential field created by an attractive target, from [10]. The agent is programmed to move in the direction with the greatest drop in potential energy until it reaches the target which has the lowest potential energy.

2.1.1 Problems with PFM

PFM is not perfect; there are at least four problem areas as discussed by Koren et al. [9]. The first, and the most referenced trouble area, is the creation of trap situations due to local minima in the potential field [8]. This problem arises when the robot finds itself in a dead-end, which can easily arise by nearing, e.g., a U-shaped obstacles or environment boundary. One possible remedy to this issue is global path planning [11]. Global path planning allows

the robot to identify that it is in a potential well trap and determine the best route to escape the trap and continue towards the goal. A second limitation is the frequent inability to travel between two obstacles even though they have sufficient physical separation that the robot should be able to pass through. This limitation can make it difficult for unmanned vehicles to travel through doorways or enter hallways. A solution to this problem is to carefully tune the relative magnitudes of the target and obstacle forces. A third limitation is oscillations in the presence of obstacles. Unstable motion can result as the vehicle travels alongside an obstacle, especially if there is a discontinuity in an otherwise smooth surface. A fourth problem is unstable oscillations while traversing a narrow corridor. While in a narrow corridor, the robot is repulsed by both sides. If the robot detects a discontinuity in the corridor, frequently the vehicle will start oscillating rapidly and eventually have a collision.

2.1.2 Examples of Swarm Robot Experimentation using PFM

Multiple research groups successfully use PFM-based frameworks to control groups of robots. For example, Barnes et al. created a framework for both simulation and physical experiment involving groups of four and ten robots involved in a convoy protection scenario [12]. This approach used potential fields calculated using distance from the convoy vehicle, distances between robots, and a minimal number of weighting parameters. This method for multi-robot has commonly been used in robotics competitions such as the RoboCup Robot Soccer League [13], [14]. As discussed in [13], potential fields are calculated based on the layout of the playing field and the objects in the field. These fields determine the destinations and actions of each of the robot soccer players.

2.2 Physicomimetics

A physicomimetics, aslo known as AP, based framework can be used to control behavior for a whole swarm of unmanned vehicles. This framework is one of several varieties of PFM-related approaches. Physicomimetics introduces artificial forces that are converted to velocity commands for unmanned vehicles. The forces are usually calculated based on a robots distance from other robots, obstacles, and way points. Artificial forces are calculated at run-time vice calculating potential fields in advance, then later converting the fields into forces. Figure 2.2 shows how the physicomimetics framework comprises the

outer control loop, which feeds velocity commands to the inner robot control loop, which translates these commands into motor and servo actuation. When this framework is applied to multiple vehicles, the robot group can self-create a formation in a similar fashion to how intra-molecular forces cause a crystal lattice to form [15]. The formation that the vehicles form will nominally be the one with the lowest potential energy. An additional friction force is usually added for stability. The mathematical basis for physicomimetis or AP is given in Chapter 3.

Physicometics is a widely implemented framework because of its adaptability and intuitive approach. In [16], artificial forces are based on solid, liquid, and gas particle interactions, which translate to different types of swarm behavior. Although usually applied to holonomic vehicles, physicometics is also useful for controlling nonholonomic vehicles such as fixed wing aircraft [17]. This is accomplished by adding nonlinear constraints, such as treating the agent as a collection of particles vice a single particle and adding the concept of torque. This approach was tested in a fixed wing swarm UAV plume detection simulation in [18] with favorable results. Apker et al. went on to test a physicometics-based framework in a four-robot auditory scene monitoring scenario [19]. In his experiments, Apker et al. uses the four-wheeled Pioneer3-AT ground robot, which requires modifications to the artificial physics algorithm to account for the platform being a very poor approximation of a point particle. In [19], four robots traverse a cluttered environment and autonomously create a diamond formation once at an attractive goal way point.

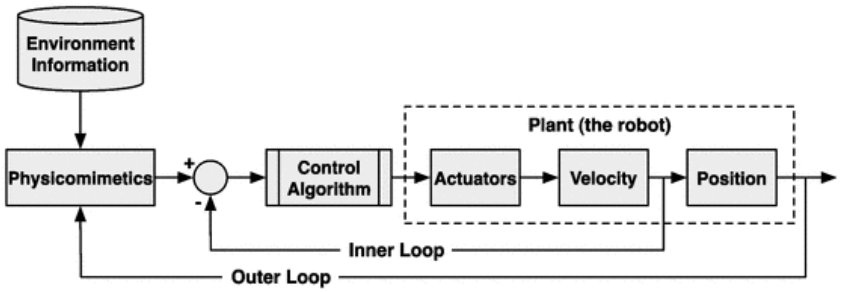


Figure 2.2: Relationship between a robot's control system and a physicometics-based framework, from [20].

There are many challenges for taking a physicometics implementation that works in simulation and applying it to physical unmanned systems. Apker and Potter [20] give several recommendations for making this transition.

2.3 Human Swarm Robot Interaction

HSRI is a previously neglected, but now fast-growing, facet of swarm research. For swarms of fairly unsophisticated robots to exhibit sophisticated group behavior, a human operator needs to inject guidance and extract information from the swarm [21]. The operator needs to be able to insert domain knowledge without needing to control individual robots. In [21], the operator controls the swarm by taking control of an individual robot or small groups of robots. The user directs the individual or small group towards an objective and the swarm follows on their own. Ideally, this is accomplished through an intuitive interface, such as a hand-held controller vice a keyboard and mouse.

Another way to control swarm behavior is to have the operator physically located among the vehicles in the swarm. The vehicles are both attracted and repulsed by the human and maintain a formation around him or her. This approach was applied to a firefighter search and rescue simulation in [22] and was named the social potential field method. Although the study did not involve real robots or human operators, it demonstrated that the social potential field method is an effective way to develop formations, avoid obstacles, navigate, and respond to the failure of individual vehicles.

2.4 Swarm Robot Platforms

2.4.1 Fixed Wing UAVs

Fixed wing UAVs are nonholonomic, and due to logistical and field testing constraints are typically employed less frequently in multi-robot research. The University of Pennsylvania GRASP Lab has successfully flown two fixed wing UAVs in a leader-follower way point following scenario [23]. Massachusetts Institute of Technology uses multiple fixed wing gasoline powered UAVs built entirely from commercial-off-the-shelf (COTS) components to test group control algorithms [24]. Stanford University's Dragonfly Project uses a ground command station (GCS) developed by Boeing to test cooperative algorithms on two fixed wing UAVs [25]. Internationally, research groups such as University of Sydny's

Australian Centre for Field Robotics and the University of Porto's Apollo project also focus on multiple fixed wing UAV research [26] and [27]. Chung et al. at Naval Postgraduate School (NPS) [28] set the goal to have 50 unmanned fixed wing UAVs fight 50 other fixed wing UAVs in an outdoor environment by 2015.

2.4.2 Rotary Wing UAVs

Swarm research centered around multi-rotor UAVs has increased due to the proliferation of relatively cheap hobby- and research-grade platforms [29]. Today, quadrotor UAVs are an increasingly common research platforms due to their simplicity and prevalence. One major limitation of small rotor-based UAVs is the limited payload and endurance. However, addressing the former concern, if multiple UAVs are deployed in a cooperative fashion, it is possible to move payloads much greater than any single UAV could handle [30]. Some research groups, such as [31], have opted to create their own in-house custom micro UAVs. One advantage of this approach is the ability to tailor the hardware and communications paths to accomplish specific research goals. Like our research group, others also utilize the Parrot AR.Drone for swarm research [32]–[34]. In most cases, the research groups only modify or create custom software for the ground control station (GCS) and do not attempt to modify the AR.Drone's firmware which is closed source.

CHAPTER 3:
Experimental Design

3.1 System Description

3.1.1 Platform

We used the Parrot AR.Drone 1.0 as our main research platform. The AR.Drone 1.0 as shown in Figure 3.1 is a quadrotor UAV with two integrated video cameras. By varying the relative speeds of its four blades, the drones moves with six degrees of freedom [35]. One 1300mAh after market, 11.1V LiPo powers the drone and gives approximately 13 minutes of flight time. AR.Drone 1.0 uses microelectromechanical systems (MEMs)-based inertial measurement units to estimate pitch, roll, and yaw. These measurements are used by the on board processor for automatic pitch, roll, and yaw stabilization. This automatic stabilization makes the AR.Drone piloting very intuitive. Based on simple commands from the user, the drone performs takeoff and landing without any manual control. In the absence of a movement command, the drone automatically trims and hovers in place, even in the presence of mild wind. The drone communicates with its controller via an ad-hoc network setup over WiFi. Control software is available for most WiFi-capable portable devices such as the iPad or Android-based mobile devices. From its inception, the AR.Drones were

Figure 3.1: AR.Drone 1.0, from [36]

designed to allow third-party developers and hobbyists to create custom software. Parrot released a series of software development kits (SDKs), which simplifies writing custom applications for controlling the drone from, e.g., a Linux desktop computer, Apple iOS device, or an Android device.

3.1.2 Robot Operating System

Robot Operating System (ROS) is an open-source framework for robot applications [37]. It functions as middleware software that provides a structured communications layer to allow various processes and/or machines with different configurations (e.g., sensors, actuators, programming languages) to interact [37]. ROS packages contain *nodes* which can send commands, process sensor information, and communicate with other nodes. For agent control, we used the `ardrone_autonomy` package [38], a ROS driver for the AR.Drone that leverages the Parrot SDK. One critical aspect of the project is developing a framework for the UAVs to self-organize and move as a coordinated swarm. Our AP based framework was implemented as several nodes within a ROS package, described further in Section 3.3.

Robot Three-Dimensional Visualization Tool (RVIZ)

RVIZ is a three-dimensional (3D) visualization tool for ROS [39]. Within RVIZ, a user can place markers that represent robots or robot components and visualize how algorithms behave in a virtual environment before testing in the real world. RVIZ itself does not have a physics engine or understanding of how the simulated robots work. However, RVIZ works with the ROS tf package, which manages coordinate frame transformations (see Section 3.1.4) and is an excellent tool for debugging reference frame dependencies. RVIZ shows marker movement in real time, which is useful for heuristically setting gains and parameters.

3.1.3 Indoor Positioning System via Motion Capture Cameras

We utilized the robot experimentation facilities available in the Naval Postgraduate School's Unmanned Systems Lab, which consists of a high-bay space equipped with a twelve-camera Vicon motion capture system [40]. The Vicon system functions similar to how a global positioning system (GPS) receiver works outdoors. Using Vicon, we can accurately track objects in a controlled indoor environment. Figure 3.3(a) shows the space when viewed from an adjacent stairway while Figure 3.3(b) shows what the Vicon cameras look like. The Vicon coverage area footprint is approximately a 10 by 15 meters (m) rectangle. The Vicon system uses near-infrared cameras to track reflective markers with sub-millimeter accuracy. To track each individual drone, we affix an unique constellation of markers to each drone's outer shell as shown in Figure 3.4(a). The Vicon system also tracks the human operator via a helmet which has its own unique marker constellation, see

Figure 3.4(b). The Vicon cameras are connected to a networked receiver which sends data to an attached computer running the Vicon software suite. Vicon software identifies and tracks all the unique constellations that are in the Vicon space, see Figure 3.2.

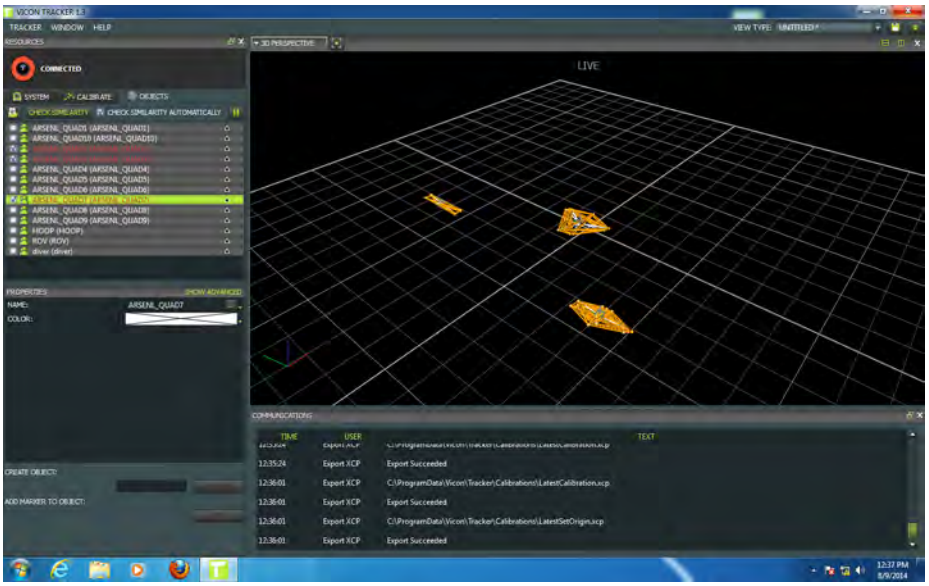
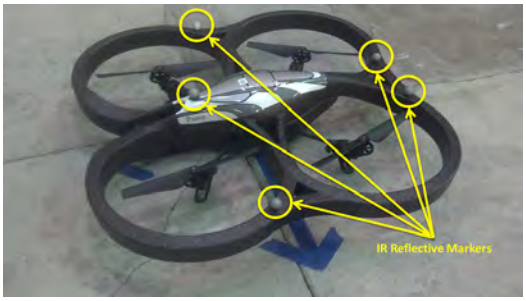


Figure 3.2: Vicon motion capture software, from [41]

We used the `vicon_bridge` package [42] to bring data from the Vicon system into the ROS environment. This package publishes translation and orientation data relative to the lab reference frame for all tracked objects in the Vicon field of view. Under nominal conditions, translation and orientation data streams at 100 Hz.



(a) AR.Drone with unique marker constellation for Vicon identification and tracking, from [41]

(b) Operator helmet with unique marker constellation for Vicon identification and tracking

Figure 3.4: Vicon marker constellations for object identification and tracking

(a) View of Vicon Space (b) Motion Capture Cameras, from [41]

Figure 3.3: CAVR Unmanned Systems Vicon Space.

3.1.4 Reference Frame Management

Reference frame management is paramount to the success of any framework that attempts to provide centralized control to multiple robots simultaneously. The speed at which the controlling station can translate between reference frames is also vitally important. In this thesis, each drone's position is referenced to the lab frame through Vicon, but each drone has its own internal reference frame to track relative positions to other drones, the human operator, and waypoints. The ROS `tf` package was used for accomplishing this task. This package provides a standardized way to track coordinate frames and transformations for an entire system of reference frames [43]. Figure 3.5 shows the hierarchy of reference frames. ROS `tf`, greatly simplifies the amount of code required for making calculations between the world and agent reference frames and eliminates errors associated with manually coding all the reference frame translation matrices. Typically, the `tf` package takes between three and four milliseconds to lookup the transform between two reference frames.

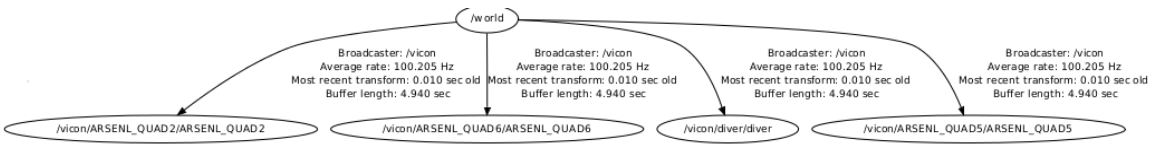


Figure 3.5: Reference frames tracked by the Vicon system, graph generated by tf

3.2 Artificial Physics Algorithm

We created a AP, or physicomemetics, based framework that ingests position and orientation information for all agents and outputs velocity commands to all agents based on their position relative to each over, way points, and the operator. The first step in this process is to turn the Vicon position data into a set of reference frames that the `tf` package can utilize. This reference frame data is used by a custom artificial physics node which stores the translations in a m by n matrix where m is the row index and n is the column index shown in Equation 3.1. To further clarify, if m equals one and n equals two, this cell holds the relative reference frame information from agent two compared to agent one. Each cell contains a quaternion ($\mathbf{q_{m,n}}$) to define a rotation and a translation vector ($\mathbf{r_{m,n}}$) used to translate from one local agent reference frame to another where i, j, k, and w are the quaternion basis elements and $\hat{x}, \hat{y}, and \hat{z}$ are the 3D unit vectors. The center diagonals are zero because these cells link one agents reference frame to its own reference frame.

$$
\begin{aligned}
T_{m,n} &= \begin{pmatrix} 0 & \mathbf{q_{1,2}}, \mathbf{r_{1,2}} & \cdots & \mathbf{q_{1,n}}, \mathbf{r_{1,n}} \\ \mathbf{q_{2,1}}, \mathbf{r_{2,1}} & 0 & \cdots & \mathbf{q_{2,n}}, \mathbf{r_{2,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q_{m,1}}, \mathbf{r_{m,1}} & \mathbf{q_{m,2}}, \mathbf{r_{m,2}} & \cdots & 0 \end{pmatrix} \\
\mathbf{q_{m,n}} &= a_{m,n} i + b_{m,n} j + c_{m,n} k + \omega_{m,n} \\
\mathbf{r_{m,n}} &= x_{m,n} \hat{x} + y_{m,n} \hat{y} + z_{m,n} \hat{z}
\end{aligned}
\tag{3.1}
$$

Using the translation vectors from Equation 3.1, we calculate the distance, $d_{m,n}$, between each agent in the $x-y$ plane using Equation 3.2, and store these data in a matrix given by Equation 3.3. Similarly, distances between each agent and waypoint as well as between each agent and the embedded human operator are calculated and stored.

$$
d_{m,n} = \sqrt{x_{m,n}^2 + y_{m,n}^2} \tag{3.2}
$$

$$
D_{m,n} = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & 0 & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \cdots & 0 \end{pmatrix} \tag{3.3}
$$

Using the distances stored in the matrix shown in Equation 3.3, Equation 3.4 calculates the artificial force, denoted $\mathbf{F_{agent}}$, felt by agent m due to agent n, where M is the virtual mass of each AR.drone, p, and G are constants. We set M to one for all agents, but this number could be increased or decreased to give an agent more or less influence over its peers. If the separation distance is greater than the desired separation distance, Equation 3.4 is multiplied by -1 to make the force attractive. Figure 3.6 is a graphical representation of the virtual force magnitude as a function of distance between agents. Structuring the virtual force in this way causes the agents to move towards each other until they reach their desired separation distance. At this distance, they begin to repel each other. Overall, this interaction causes the agents to maintain the desired separation distance. Using the distances between each agent and the operator ($d_{m,op}$), Equation 3.6 calculates the artificial force ($\mathbf{F_{operator}}$) felt by agent m due to the operator, where p and C are constants, while the distance between the drone and operator is less than twice the desired separation distance. If the separation distance is greater than the desired separation distance between the operator and the agent, Equation 3.5 is multiplied by -1 to make the force attractive, see Equation 3.6. If the distance between the operator and the agent is greater than twice the desired separation distance, Equation 3.7 is used to calculate $\mathbf{F_{operator}}$. This decreases the amount of time it takes the drones to find the operator if they are initially far away.

$$\mathbf{F_{agent}} = \frac{GMM}{(d_{m,n})^p}\hat{\mathbf{r}}_{\mathbf{m,n}} \tag{3.4}$$

$$\mathbf{F_{operator}} = \frac{CM}{(d_{m,op})^p}\hat{\mathbf{r}}_{\mathbf{m,op}} \qquad \text{where} \qquad d_{m,op} < d_{desired} \tag{3.5}$$

$$\mathbf{F_{operator}} = \frac{-CM}{(d_{m,op})^p}\hat{\mathbf{r}}_{\mathbf{m,op}} \qquad \text{where} \qquad d_{desired} < d_{m,op} < 2d_{desired} \tag{3.6}$$

$$\mathbf{F_{operator}} = -CMd_{m,op}\hat{\mathbf{r}}_{\mathbf{m,op}} \qquad \text{where} \qquad d_{m,op} \geq 2d_{desired} \tag{3.7}$$

Figure 3.7 shows force felt by the agent as a function of separation distance from the operator. The force profile is designed to allow the agent to quickly close distance to the operator and maintain the desired separation. Using the distances between each agent and the current way point, that is, $d_{m,wp}$, Equation 3.8 calculates the artificial force, $\mathbf{F_{waypoint}}$, felt by agent m due to the current way point where B is a proportionality constant. Unlike

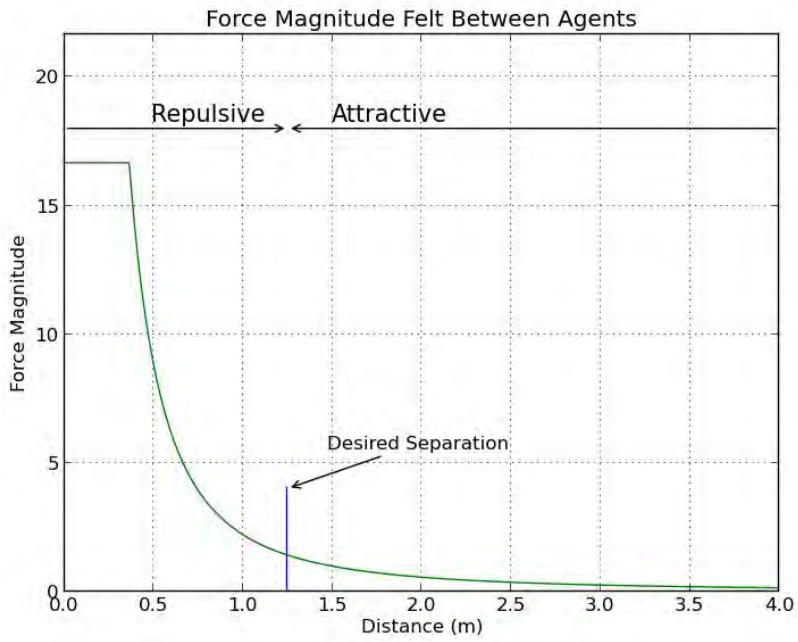


Figure 3.6: Virtual force magnitude felt by agent m due to agent n ($G = 2.2$, $p = 2.0$), after [15]

the virtual forces due to the operator and other agents, the force felt due to the way point is always attractive. Figure 3.8 is a graphical representation of the virtual force magnitude as a function of distance between agent and way point.

$$\mathbf{F_{waypoint}} = -B(d_{m,wp})\hat{\mathbf{r}}_{\mathbf{m,waypoint}} \tag{3.8}$$

In Equation 3.9, the force vectors are summed to give a net resultant force vector.

$$\mathbf{F_{m,net}} = \mathbf{F_{waypoint}} + \mathbf{F_{operator}} + \sum_{i=1}^{n} \mathbf{F_{agent}} \tag{3.9}$$

From $\mathbf{F_{net}}$ we calculate the velocity vector $\mathbf{v}$ using Equation 3.10

$$\mathbf{v}_m = \frac{\mathbf{F_{m,net}}\Delta t}{M} + \mathbf{v}_{m,0} \tag{3.10}$$

where Δt is the amount of time between velocity update cycles and $\mathbf{v}_{m,0}$ is the velocity vector from the previous time step. In our algorithm $\mathbf{v}_{m,0}$ is reset to zero each time step.

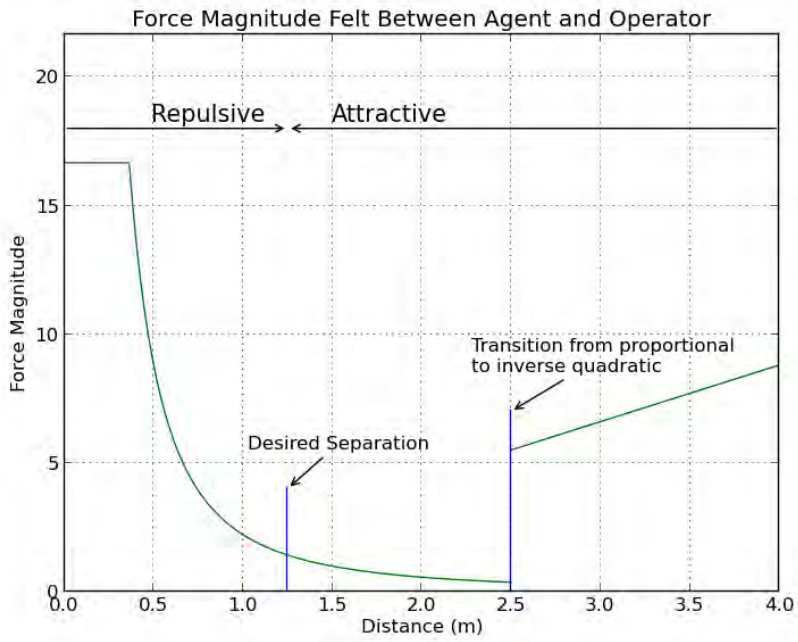


Figure 3.7: This curve describes the force Magnitude felt by the agent due to proximity to the human operator ($C = 2.2,\ p = 2.0$)

This effectively turns off Newton's first law of motion: "An object at rest will remain at rest unless acted on by an unbalanced force. An object in motion continues in motion with the same speed and in the same direction unless acted upon by an unbalanced force." In our framework, an object or AR.Drone must be subjected to a virtual force to move, otherwise it stops. The motivation for this was to keep fast moving drones from leaving the Vicon coverage area. The velocity vector is broken into its x, y, and z components which are sent to the agents as commands. We set the parameters G, C, and B such that commanded drone speed would be below 0.3 meters per second (m/s) based on the size of the arena and the maximum separations possible.

3.3 System Integration

Running a simple AP experiment involves multiple components working together. Figure 3.9 shows a simplified system architecture. The Vicon camera system and attached computer detect and track objects inside the Vicon space. This information is made available on a server stack. The controlling station is typically a Linux-based laptop running

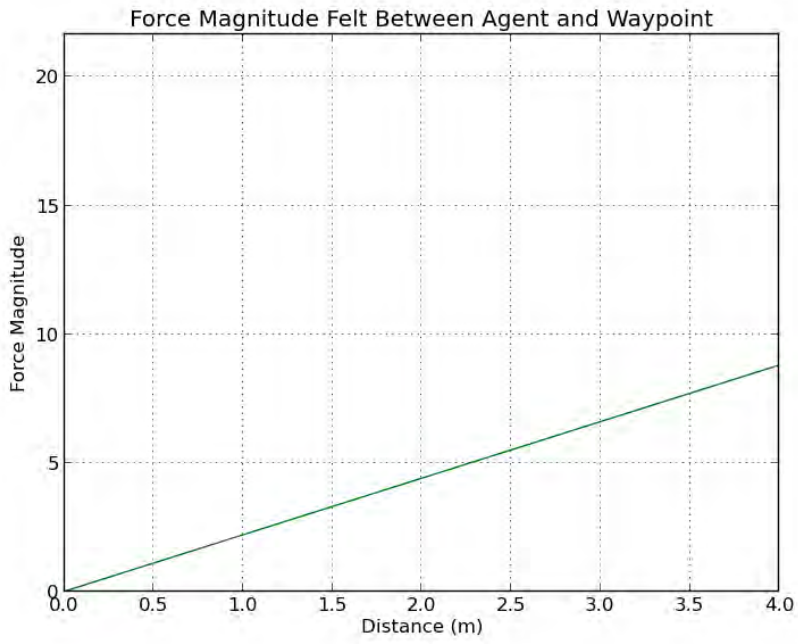


Figure 3.8: This curve describes the force magnitude felt by the agent due to proximity to the current way point (B = 2.2). Unlike the other two force magnitude relationships, force magnitude due to distance from way point is linear.

Ubuntu 12.04, ROS version Groovy, and Python version 4.6.3. The controlling station runs all the ROS nodes for the experiment and controls each of the drones through the server's attached wireless router.

3.3.1 Drone Communication

Out of the box, the AR.Drones are not designed to have multiple drones connected to one controller. To overcome this limitation and to control all the agents from one station, NPS researcher Michael Clement [44] created an automated WiFi scanning tool to enable telnet access into each detected drone and connect it to a wireless router connected to a server under a predefined IP address scheme shown in Table 3.1. By default, the drones are set to use the same user datagram protocol (UDP) network ports. This led us to create a UDP remapping tool to allow one computer to control all the drones. An explanation of how Clement's multiple AR.Drone communications mapping is available at [44]. Both the controlling station and the Vicon system are connected to the drone server via ethernet. This

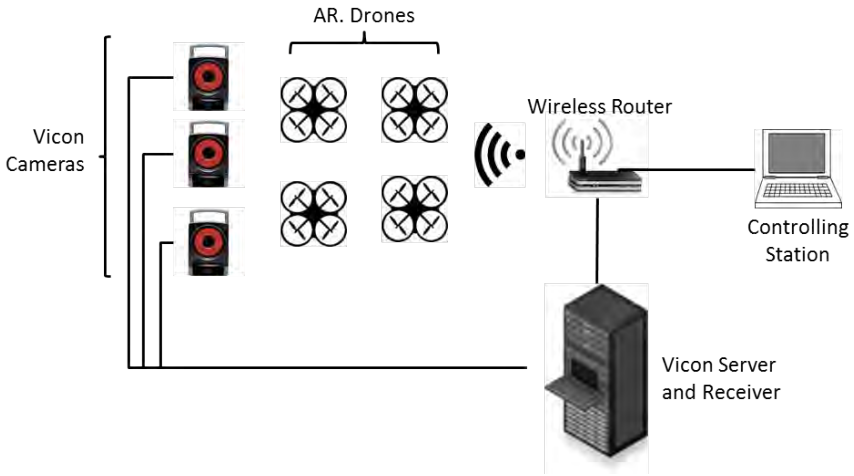


Figure 3.9: Hardware architecture consists of the integration of a Vicon camera system, multiple AR.Drones, a wireless network, and a computer functioning as a GCS

allows the controlling station to send commands to the drones and receive Vicon data using one network adapter, as illustrated in Figure 3.9. We created a ROS node called `SwarmLoad` to scan the server for connected drones and attempt to load the `ardrone_autonomy` driver for each drone. We also created a custom graphical user interface to perform group takeoff, landing, and emergency reset and to monitor battery levels, see Figure 3.11. The `vicon bridge` node gives the controlling station the positions and orientations of all the agents in the arena as well as the position and orientation of the operator. Once the UAVs are flying, the operator can start the custom AP node. Typically, either the way point force coefficient

Side Number	ROS namespace	Wireless ID	IP Address
01	Quad1	ardrone_197467	192.168.0.101
02	Quad2	ardrone_198504	192.168.0.102
03	Quad3	ardrone_258674	192.168.0.103
04	Quad4	ardrone_256959	192.168.0.104
05	Quad5	ardrone_266111	192.168.0.105
06	Quad6	ardrone_266678	192.168.0.106
07	Quad7	ardrone_198307	192.168.0.107
08	Quad8	ardrone_150853	192.168.0.108

Table 3.1: Identifiers for each AR.Drone in the swarm, from [41]. This table connects and identifies each drone with a hull number, ROS namespace identifier, SSID (Wireless ID), and IP number.

or the operator force coefficient are set to zero. This either places the experiment in operator follow mode or way point follow mode, respectively.

3.3.2 Software Architecture

Figure 3.10 shows the software architecture for agent tracking, receiving commands from the graphical user interface (GUI) and the hand-held controller, implemented on a Nintendo Wiimote (see Section 3.3.3), and sending commands to the agents.

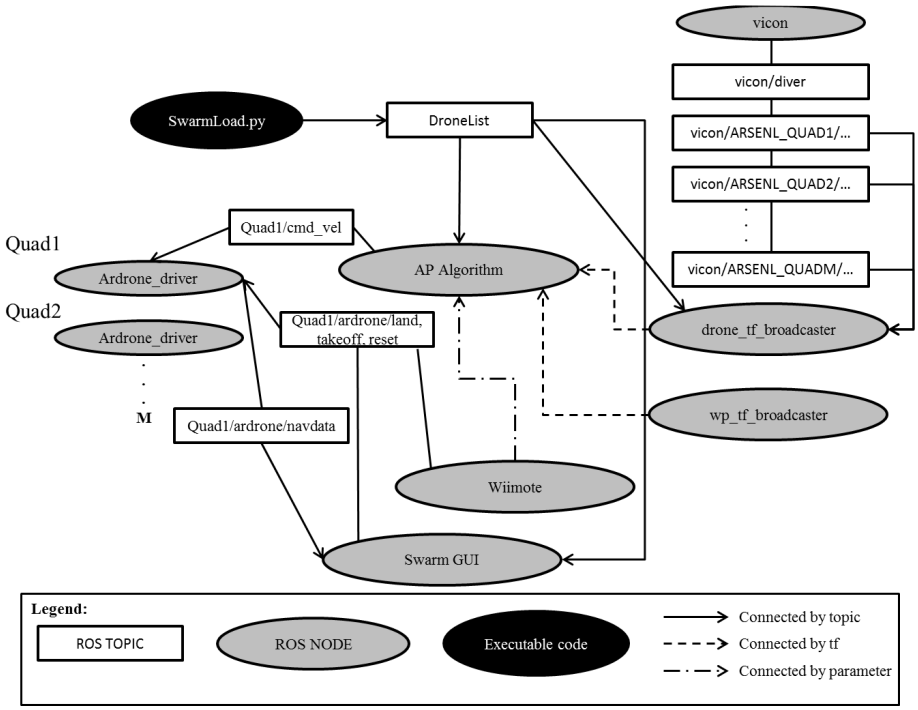


Figure 3.10: Simplified view of software architecture, for a more complete software architecture see (Appendix A

)

`SwarmLoad.py`

`SwarmLoad.py` is an executable Python script that scans the Vicon server for connected AR.Drones. The script pings each drone's network IP address based on the known IP address scheme, as defined in Table 3.1. If there is less than 100% packet loss (meaning positive communication with the drone is achieved), the script attempts to start an instance of the AR.Drone driver. `SwarmLoad.py` provides a set of parameters to each driver that

control: port addressing; video transmission rate; maximum altitude; minimum altitude; and altitude sonar pinger frequency. The AR.Drone has two sonar pinger frequencies available. If multiple drones use the same frequency, then interference is likely, which results in altitude control problems. `Swarmload.py` addresses this limitation by alternating the pinger frequencies. For example drone one will operate using frequency A, drone two will operate on frequency B, but because there are only two frequencies available, drone three will also operate on frequency A. After starting an instance of AR.Drone driver for each connected drone, `Swarmload.py` transmits a list of the drones using a ROS topic called `DroneList`. This topic informs the other nodes in the software architecture how many and which AR.Drones are involved in the experiment.

AP Algorithm

At the core of the software architecture is the `AP_Algorithm` node. This node takes in `tf` reference frame data from the `wp_tf_broadcaster` and `drone_tf_broadcaster` nodes. Using this data, the node calculates forces as outlined in Section 3.2. These forces are converted to velocities which are published to each of the `ardrone_driver` nodes using the `cmd_vel` topic.

Vicon

The `vicon_bridge` node publishes to a topic for each drone in the arena (`vicon\ARSENLQUAD#\ARSENLQUAD#` where # is the drone's identification number) and to a topic associated with the operator (`vicon\diver\diver`. These topics contain position and orientation data. Position is given in the form of a translation vector and orientation is represented by a quaternion. These data are converted into `tf` reference frames by the `drone_tf_broadcaster` node.

3.3.3 Human Swarm Robot Interaction

A user also has basic takeoff, land, and emergency reset control of the drones via a custom graphical user interface (GUI), as illustrated in Figure 3.11. The push buttons are configured to send `Takeoff`, `Land`, and `Emergency Reset` commands to the selected drones, indicated by checked boxes. For example, when the `Takeoff` button is pressed and activated, the GUI publishes a message to the `Quad#\ardrone\takeoff` (where # is the drone's identification number) topics for each drone with a checked box. After the drones

takeoff, the *Drone Status* for each drone changes from `Landed` to `Flying` based on information received from the `Quad#\ardrone\navdata` topics. In a similar fashion, the GUI receives battery charge information from the `Quad#\ardrone\navdata` topics and displays the percentages in the window.

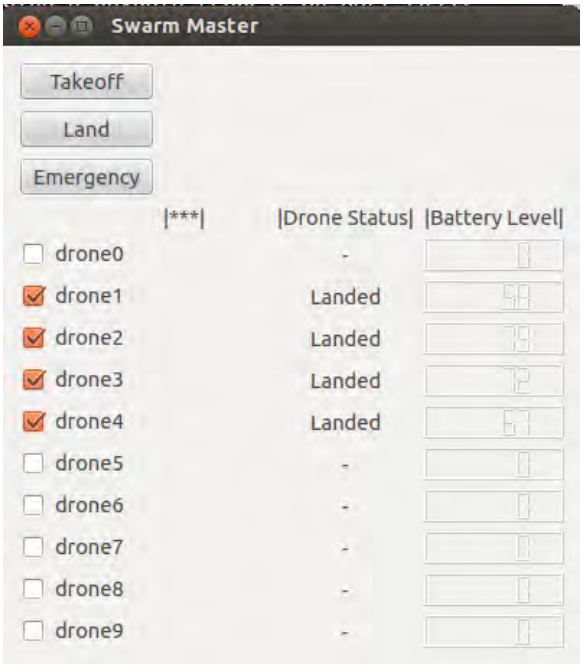


Figure 3.11: Custom GUI for basic group control of drones

In order to give the human operator some basic control over the swarm, a Nintendo Wiimote controller is integrated into the experiment. Wiimote controller state is monitored using the open source ROS wiimote package [45]. The Wiimote gives the operator the ability to collectively order swarm takeoff, landing, or emergency reset. It also gives the operator the ability to turn on and off the AP node by changing a ROS parameter used by the `AP_Algorithm` node. When the AP node is off, the drones go into hover mode. The "1" and "2" buttons respectively increase and decrease G, that is, the constant associated with the force felt by the drone due to the operator given in Equations 3.6 and 3.7. This interaction further allows the operator to heuristically tune and set this parameter. Eventually, the Wiimote is envisioned to also give the operator the ability to take manual control of individual drones and fly them traditionally, as necessitated by the mission of interest. The current implementation of the Wiimote button mapping is shown in Figure 3.12.

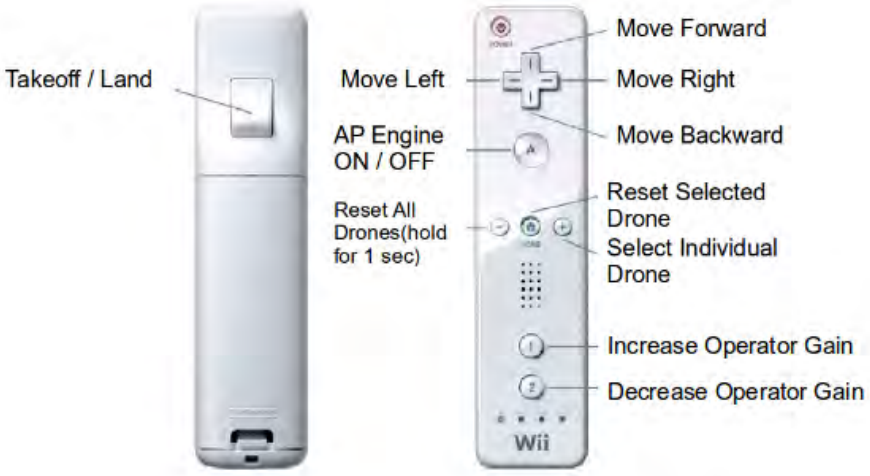


Figure 3.12: Nintendo Wiimote button mapping (manual control move left, right, forward, backward not yet implemented)

CHAPTER 4:
Results

4.1 Simulation

We created two simulation environments to test and gain familiarity with our AP based framework. The first environment consists of a simple two-dimensional (2D) playing field with point particles. For the second, we used a 3D environment, with agent orientation taken into account, and used RVIZ to visualize in real time the creation of the swarm formation lattice.

4.1.1 2D Simulations

For the simple 2D simulation, we adapted a simulation program from Spears [15]. It consists of a simple 2D playing field with a set number of point particle agents where each is given an initial random positions. For simulation visualization we used the popular `matplotlib` library [46]. Each time step the agents move a set amount based on their proximity to all the other agents. After hundreds of time steps, the agents self organize into a lattice as seen in Figure 4.1 with four agents and in Figure 4.2 with 25 agents. 2D simulation results were consistent with [15] which shows after thousands of time steps agents will create an organized hexagonal lattice (see Figure 4.3).

4.1.2 3D Simulations

We also created a more advanced three dimensional simulation using RVIZ for ROS [39]. In this simulation, the agents have orientation, and the forces are translated into the agent reference frames, see Figure 4.4. Once again, consistent with [15], the agents self-organize into a lattice after several thousand time steps. In the 3D simulation, there is no attempt to control drone altitude. All virtual forces and velocity commands are calculated in the xy plane so the lattice formation that appears is a 2D lattice similar to the results of the 2D simulations. We were able to later leverage this RVIZ code for real-time flight playback in RVIZ, which helped with troubleshooting the AP algorithm, see Figure 4.5.

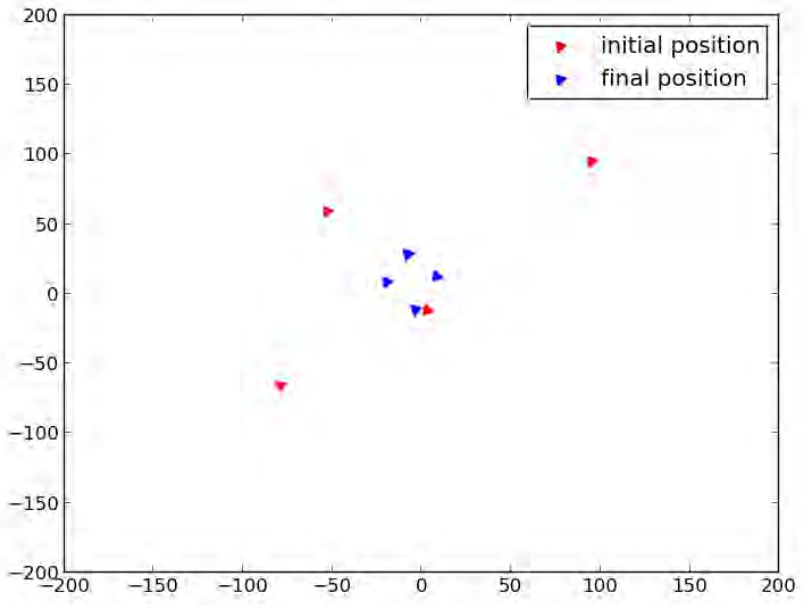


Figure 4.1: Simple 2D simulation involving four agents

4.1.3 Simulation Discussion

These simulations validated the AP based framework and demonstrated that, given enough time steps, a group consisting of a few to many agents could self-organize into a lattice by responding to AP based virtual forces. One metric for evaluating the performance of the algorithm is the amount of time it takes for all agents to position themselves at the desired separation distance from their neighbors. In the three dimensional RVIZ-based simulation, convergence typically took between two to three thousand time steps, or 20 to 30 sec, for the average separation distance between four agents to approach the desired separation distance of 1.25 m as seen in Figure 4.6.

4.2 Physical Experimentation Results

Our research group encountered multiple challenges during implemention of the framework on the actual AR.Drones. These challenges included networking, Vicon operating system compatibility issues, `tf` implementation problems, and drone altitude sonar pinger interference to name several hurdles. Eventually, we implemented the AP framework on

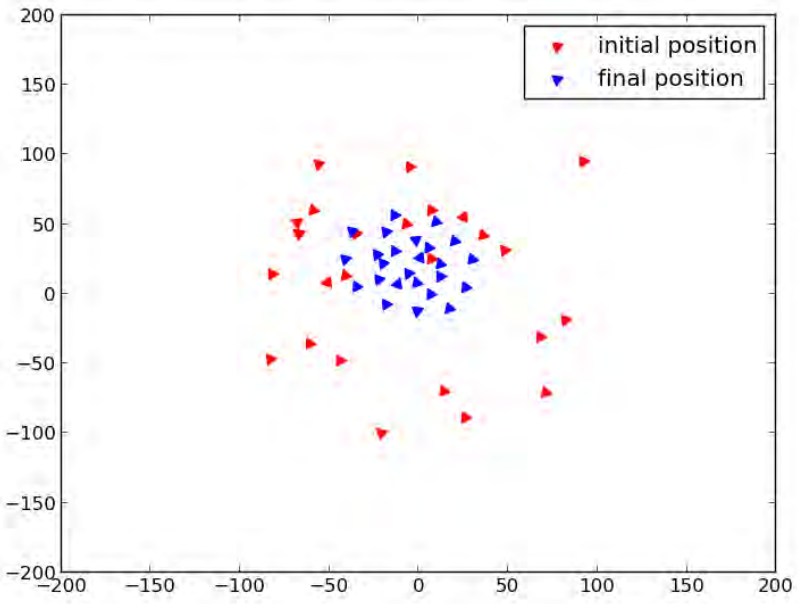


Figure 4.2: Simple 2D simulation involving 25 agents

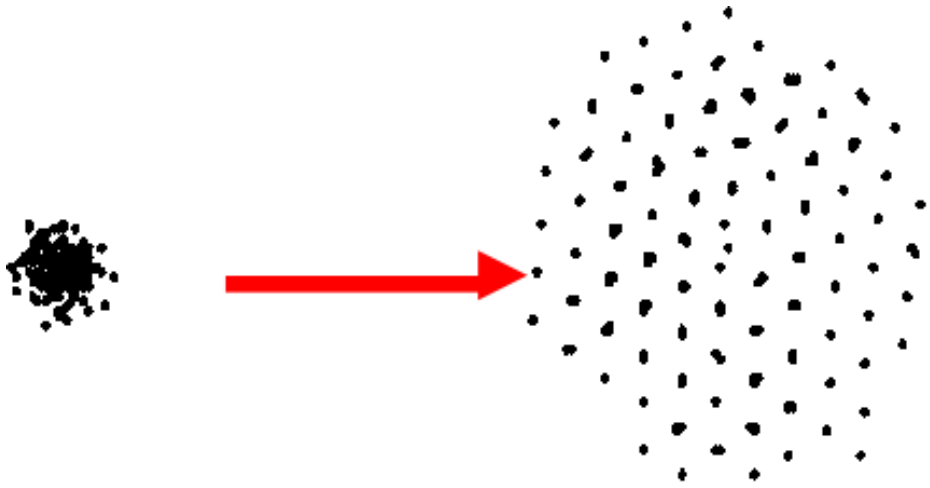


Figure 4.3: Simple simulation run by Spears involving many agents and thousands of time steps, after [15]

three drones alongside a human operator with a Wiimote controller, as shown in Figure 4.7.

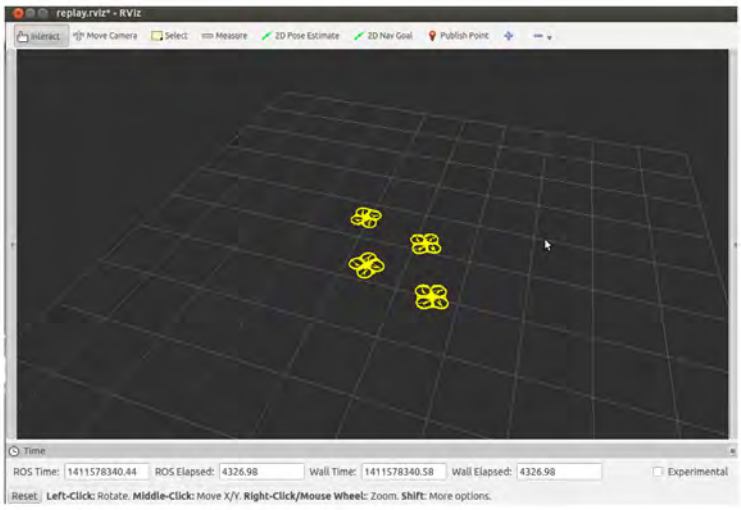


Figure 4.4: Screen Capture of AP algorithm controlling four simulated drones using robot 3D visualization tool, RVIZ, for ROS

4.2.1 AP Framework Drone Trajectories

Figure 4.8(a) shows *xy* plane trajectories for a typical experimental run involving three drones and one human operator. In this run, the operator stayed as stationary as possible (operator translation < 4 cm as measured by Vicon) and the drones started several meters away in arbitrary positions and orientations relative to the operator. The coefficient associated with way point attraction, B from Equation 3.8, was set to zero so that the drones were only affected by other drones and the operator. After 12 seconds, all drones are within 20% separation error compared to desired separation from the operator which correlates to all drones being within 25 cm of their desired position, see Figure 4.8(b). Figure 4.9 represents a snapshot of drone and operator positions with artificial force vectors superimposed. These artificial force vectors represent the force vector summations for each drone. For example, the blue vector for drone 2 represents the summation of the following forces: attraction to the operator, attraction to drone 5, and attraction to drone 6 (as previously stated the parameter associated with way point attraction was set to zero so no attraction was felt toward the way point). Drone 2 is outside the desired separation distance to drone 5, drone 6, and the operator. Because of this, its force vector points approximately towards the operator because of the relative positions of the drones and the higher force magnitude parameter associated with operator attraction. A similar explanation could be made for

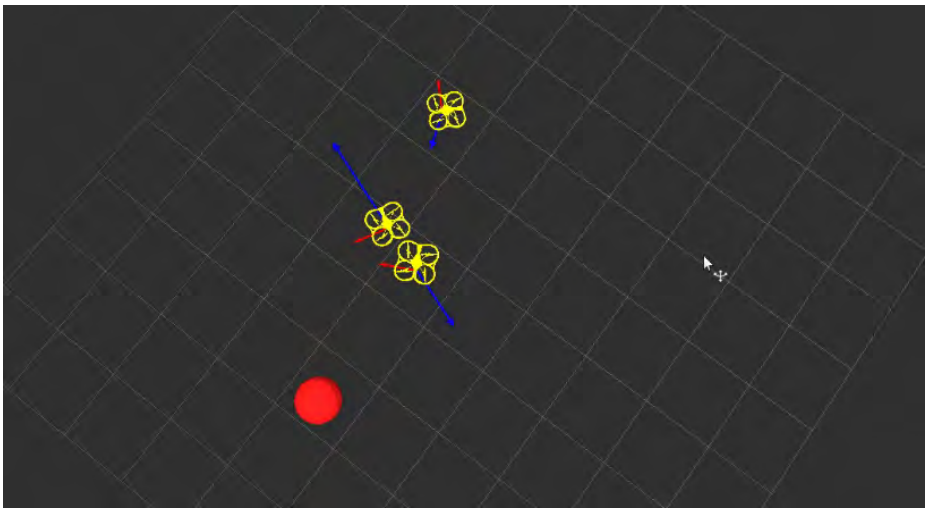


Figure 4.5: Screen capture of flight data playback using RVIZ. The red sphere shows operator location, red arrows indicate drone foward facing camera orientation, and the blue arrows represent the virtual force vectors.

drone 5's force vector. Drone 6 is within the desired separation distance of the operator which generates a strong repulsive force which is decreased due to drone 6's attraction to drone 2 and 5. Section 4.2.2 goes into greater detail on how these artificial force vectors change over time for each drone.

Figure 4.10(a) depicts the same run but several seconds later when the operator is walking in an approximately straight line. The drones are initially able to stay within 50% separation error compared to desired separation from the operator which correlates to being within 63 cm of the desired position, but after the operator stopped the drones had difficulty regaining the correct separation distance, see Figure 4.10(b). One possible explanation for this is that we did not collect enough data and that given more time the drones would have formed a tighter formation. Unfortunately, at 22 sec into the run the operator started moving again so the data does not exist to show that a tighter formation would have formed. It is also possible that system latency was higher during this part of the experiment and force calculations were being performed on position data that was not current. As discussed in Section 4.2.3, this experimental setup warranted further study of how operator position perturbations affect the drones' ability to reform into a lattice formation.

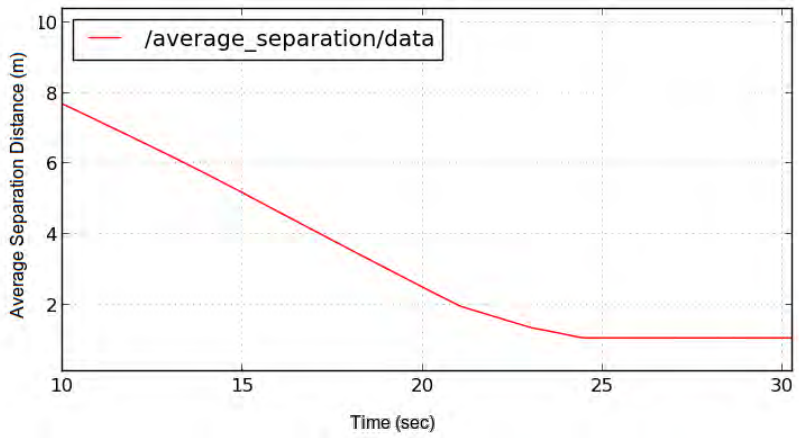


Figure 4.6: Average separation versus time with four drones for 3D RVIZ simulation (Desired Separation = 1.25, $G = 2700$, $w = 1.0$, $p = 2.0$, $\Delta t = 0.01$)

4.2.2 Artificial Force Vector Evolution

To further illustrate how the force vectors evolve over time, Figure 4.11 depicts drone position with drone force vectors superimposed over a six second time period. During this time period, the drone start in arbitrary positions, but after six seconds end up in an approximate lattice shape. After six seconds, the force vector magnitudes are much smaller than at the beginning of the experiment. This is because the repulsive and attractive forces are summing to a number near zero. Having a force vector magnitude near zero when in tight formation is desirable, because it minimizes the amplitude of oscillations, thereby decreasing the probability of a collision. Note that because of the difference between the parameters C and G, the vectors may not appear to have the correct direction or magnitude. We re-checked our results by having the AP algorithm output intermediate results to the operator's console, which we compared to hand calculations performed using raw data from Vicon. Our manual calculations were consistent with the intermediate and final results. This demonstrates that the AP algorithm was correctly implemented and that there were not code problems involving reference frame transformations or force calculations. If we had picked different force parameters, the magnitudes and directions of the resultant force vectors would be different.

The operator was able to walk around the Vicon space while the drones followed and re-formed a lattice shape around the operator when he stopped. Latency in the system and

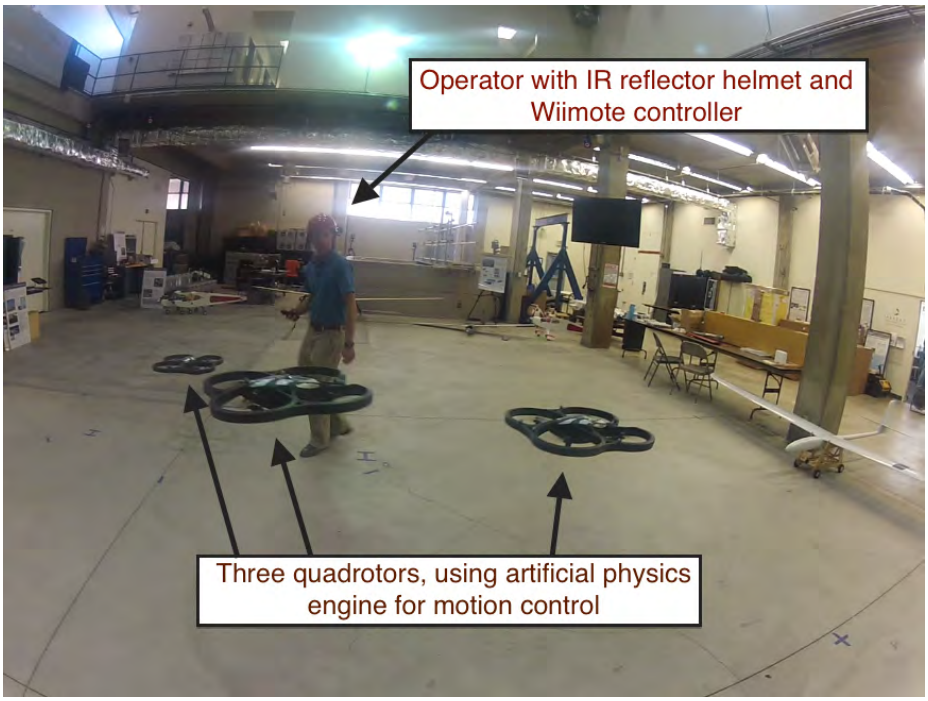


Figure 4.7: Human operator wearing Vicon marker helmet controlling three agents using Wiimote hand-held controller

imposed velocity constraints made it so the operator could escape the formation center by walking briskly. We also conducted experiments without the operator where the drones were given a set of way points to navigate but did not perform rigorous data collection on these experiments. Initial observations showed that when the a new way point was inserted, the drone would break formation, travel to the new way point, and recreate a lattice formation. We need to collect data from these way point experiments as further discussed in Chapter 5.

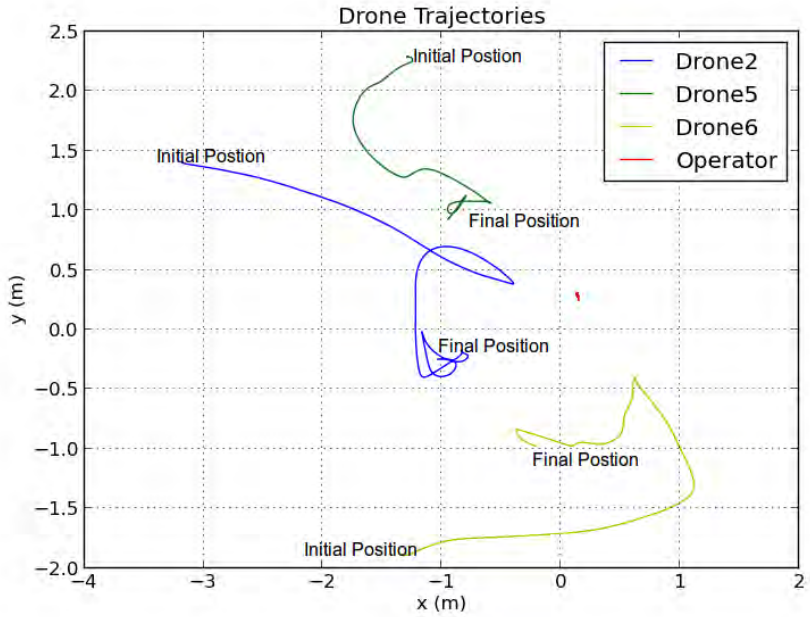
4.2.3 Additional Experiments

We ran additional experiments to examine the amount of time required after an operator position perturbation for the drones to regain desired separation. Figure 4.12 depicts the results of this experiment, and Figure 4.13 shows time-lapse photographs of the operator's position relative to the drones. The operator took a brisk one meter step in the x-direction, stayed stationary for seven seconds, and then took a brisk step to return himself to his original position. Error in average separation distance grew during and after the initial step

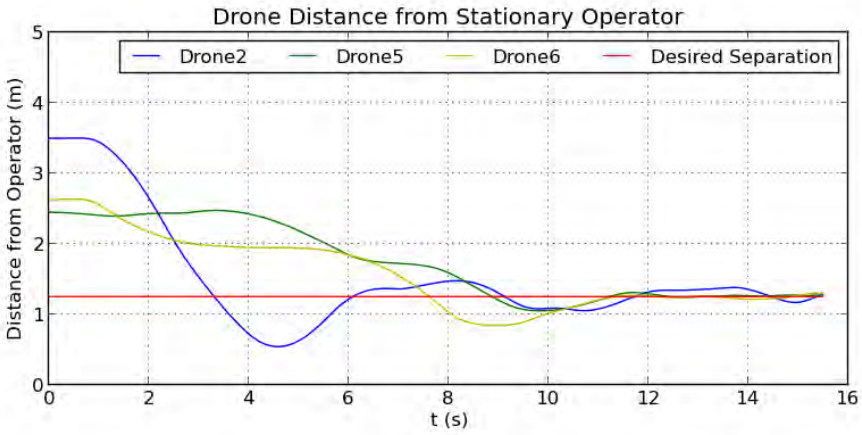
input displacement, but returned to within 25% of the desired separation distance after the perturbation was over. This experimental run exhibited a higher error in error in desired separation distance compared to the run described in Section 4.2.1. Again, this may be attributable to system latency increasing with time. It could also be due to slightly larger operator position perturbations when the operator was trying to stand still (10 cm vs. 4 cm as measured by Vicon).

4.2.4 Physical Results Discussion

Our framework was effective at autonomously organizing a group of AR.Drones in arbitrary starting positions and orientations around a human operator. The drones were fairly consistent in finding the operator and creating a formation. Even with a moving operator, the drones would follow and reform into a formation after the operator stopped. Despite these initial positive results, drone on drone collisions were not uncommon. It appears that latency in the system sometimes allows drones to get too close to each other. Once the drones are within approximately 0.5 m of each other, rotor blade wash frequently causes instability. Sometimes the drones recover, but other times the wash leads to a collision and one or both drones crash into the ground. AR.Drones are built with the home hobbyist in mind and are built to withstand multiple crashes. We frequently had to glue back on the Vicon constellation marker balls due to them falling off after a crash.



(a) xy trajectories for three drones with stationary operator



(b) Average separation distance from operator over time with stationary operator

Figure 4.8: Experimental results for three drones with a stationary operator (Desired Separation $= 1.25$ m, way point parameter $B = 0$, agent to agent parameter $C = 2.2$, agent to operator parameter $G = 3.1$, distance exponent parameter $p = 2.0$, time duration per time step $\Delta t = 0.03$ s)

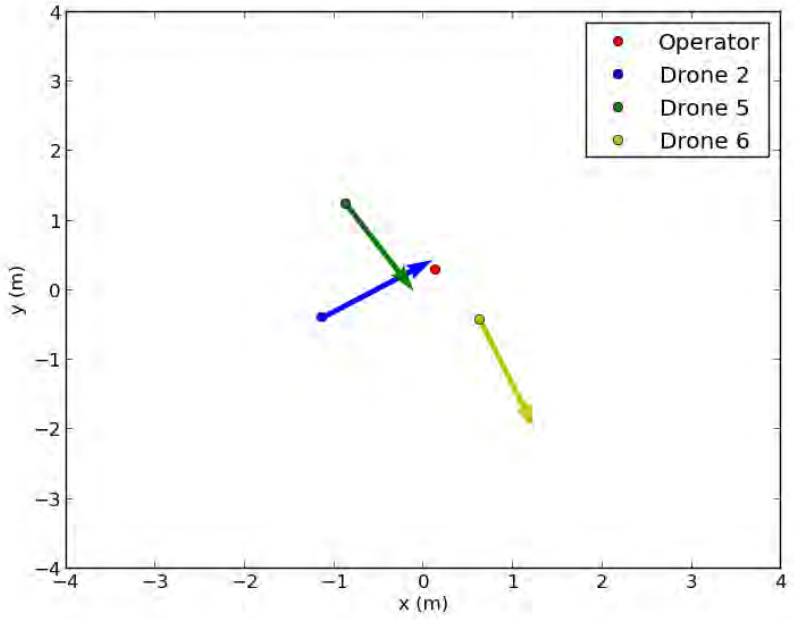
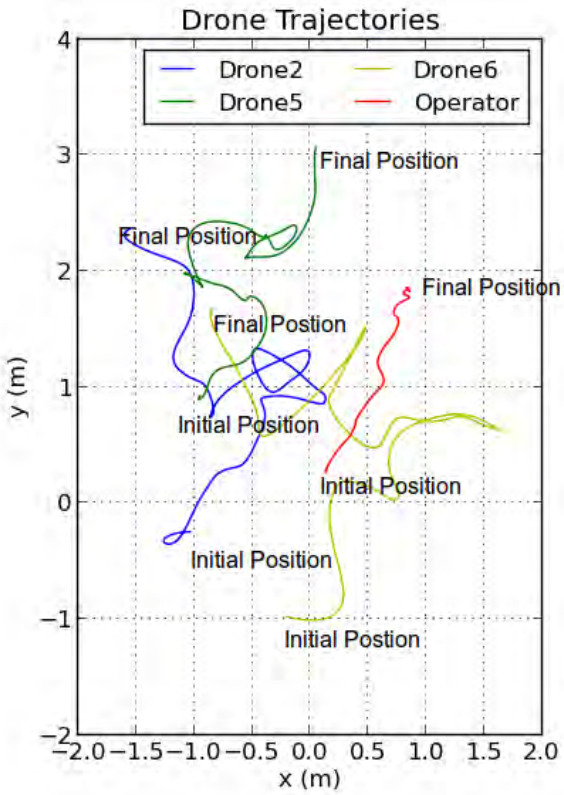
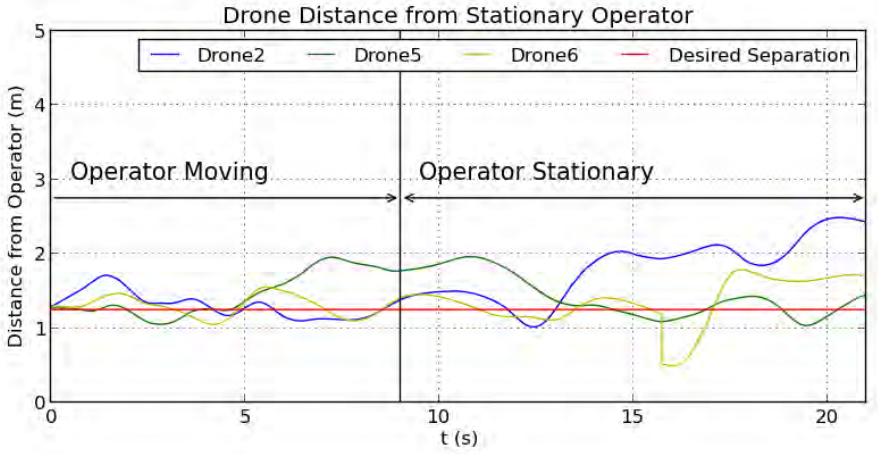


Figure 4.9: Drone and operator positions with artificial force vectors



(a) xy trajectories for three drones with moving operator



(b) Average separation distance from operator over time with initially moving then stationary operator

Figure 4.10: Experimental results for three drones with an initially moving then stationary operator (Desired Separation $= 1.25$ m, way point parameter $B = 0$, agent to agent parameter $C = 2.2$, agent to operator parameter $G = 3.1$, distance exponent parameter $p = 2.0$, time duration per time step $\Delta t = 0.03$ s)

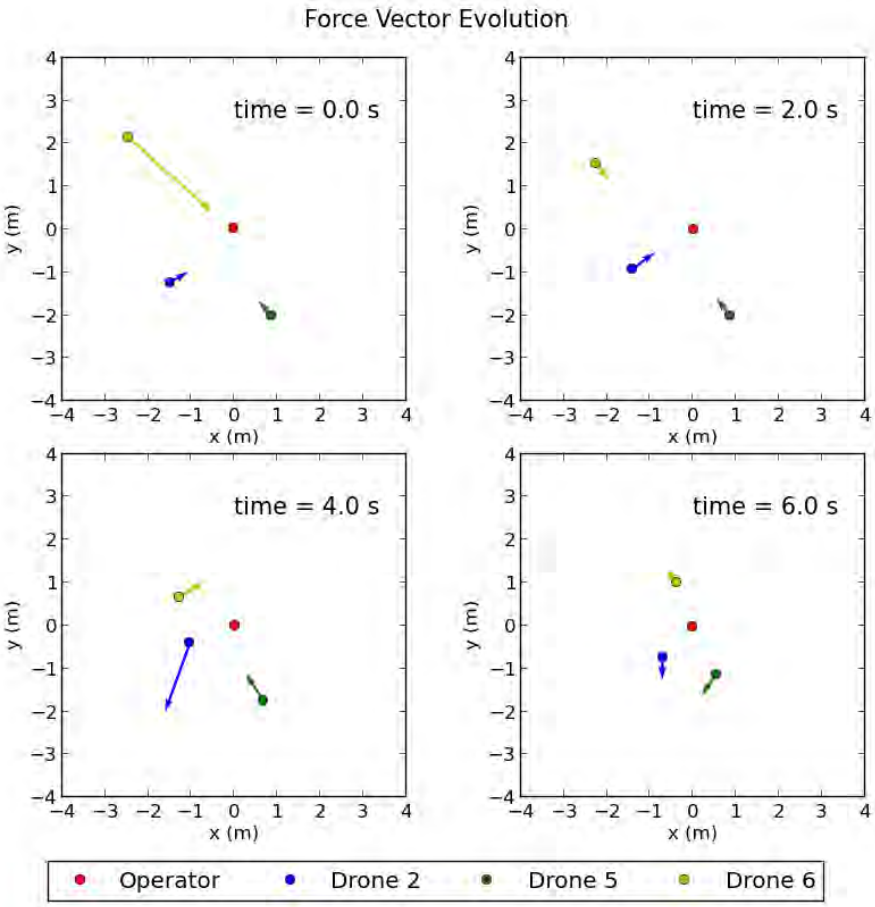


Figure 4.11: Evolution of Force Vectors over a six second time period with stationary operator. Overtime, force vectors change direction and magnitude as the geometry of the experiment changes. After the six seconds shown, the drones are in a stable formation with separation distances from the operator within 20% of the desired value (Desired Separation $= 1.25$ m, $B = 0$, $C = 3.2$, $G = 2.2$, $p = 2.0$, $\Delta t = 0.03$ s)

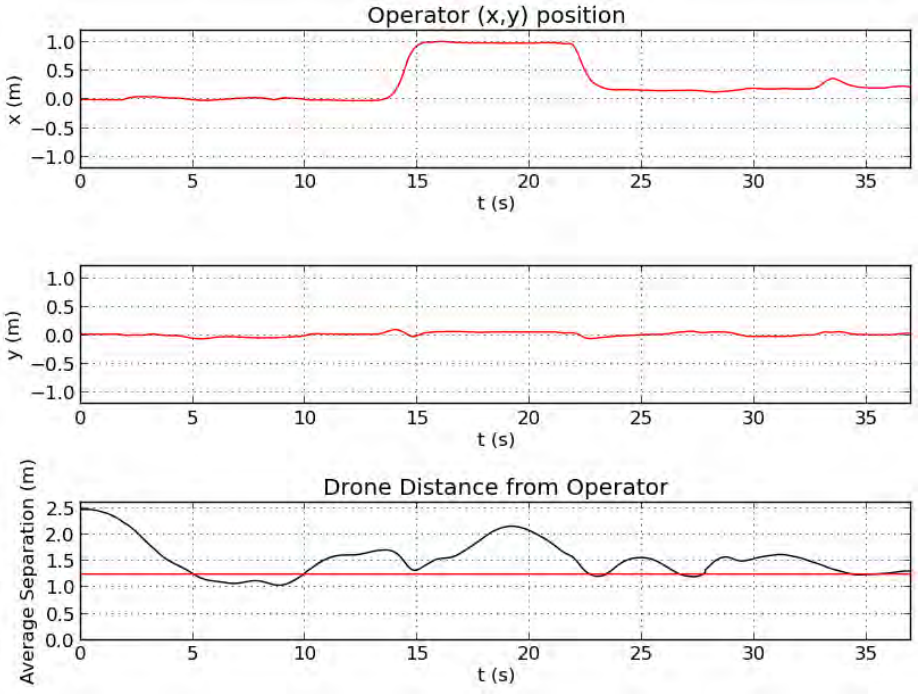
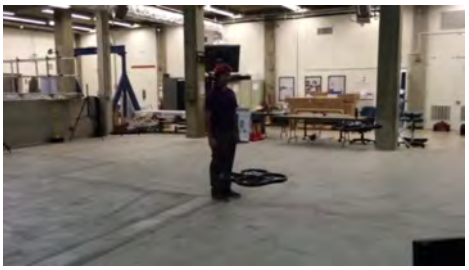
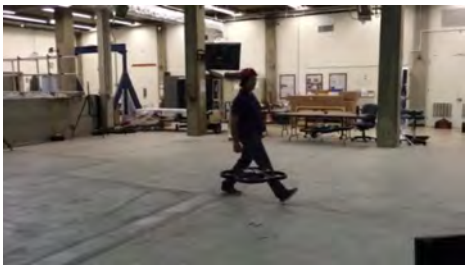


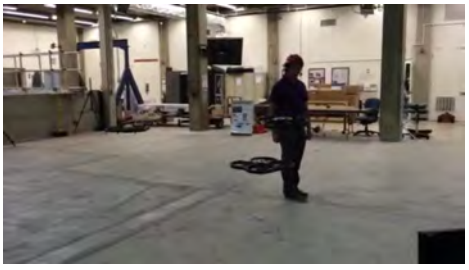
Figure 4.12: Graphs show operator discplacement in the x and y directions as a function of time. In this experiment, the operator took a one m step in the positive x direction, stood still for sevon sec, and took a step in the negative x direction to return to the origin. During the experiment, average drone to operator separation was plotted as function of time. While the operator is moving, average separation distance increases as the drones attempt to recreate the formation. After the operator returns to the origin and stops moving, average separation distance approaches the desired separation distance as the formation is recreated (Desired Separation $= 1.25$ m, $B = 0$, $C = 3.2$, $G = 2.2$, $p = 2.0$, $\Delta t = 0.03$ s).



(a) t = 10 sec

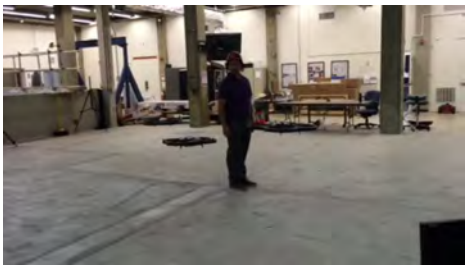


(b) t = 14 sec



(c) t = 18 sec

(d) t = 23 sec



(e) t = 25 sec

Figure 4.13: Operator and drone photographs for numerical results depicted in Figure 4.12. As the operator took a step forward the drones followed and attempted to recreate a formation.

CHAPTER 5:
Conclusion and Future Work

5.1 Conclusion

We validated our framework using both 2D and 3D simulations and successfully applied it to a group of drones which allowed them to follow a human operator inside an instrumented space. Other than initiating their collective launch with a takeoff command to the drones, no further human input was required for the drones to find and follow the operator, based on data from the indoor motion capture system. Experimental results showed that after operator movement, it would typically take between five to ten sec for the drones to form into a stable formation. A hand-held controller gave the operator the ability to takeoff and land the drones, as well as to change the coefficients associated with drone to operator attraction or repulsion forces.

In addition to the development and demonstration of the presented artificial physics-based framework, the following items highlight various contributions accomplished in this thesis:

- Development of real-time simulations in 2D and 3D environments
- Design and software implementation of `SwarmLoad.py` utility for automation of AR.Drone driver loading for multiple drones on the GCS computer
- Design and development of a dashboard GUI for basic AR.Drone group control and monitoring
- Testing and development of code for Wiimote hand-held controller integration
- Closed-loop integration and documentation of motion capture system with Robot Operating System-based software architecture
- Experimental dataset generation of multi-drone flight data for further playback and analysis

As noted in Chapter 2, there has been little prior research involving multiple UAV intentionally operating in close proximity to a human operator, although there is increasing interest in this topic. This thesis offers a possible approach for allowing humans and UAVs to safely work together in a constrained environment.

5.2 Recommendations

Given this emerging area of research, there are numerous avenues for immediate activities that will enhance further near-term developments for studying interactions between multiple drones and their operator(s).

5.2.1 Definition of Relevant Performance Metrics

This work highlighted the need to develop more rigorous metrics of performance for swarm behaviors. Tracking deviation from desired separation distance from the operator is only sufficient for assessment of a small numbers of drones. For example, if there are more than six drones, due to the nature of hexagonal lattices, there will be one drone between the seventh drone and the operator. This means that by nature of the AP algorithm, not all drones may be able to approach the desired separation distance from the operator. However, all drones should be able to reach the desired separation distance from all other drones.

As discussed by Spears [15], one method for evaluating formation quality is to average the error in the expected angle between agents within the formation lattice. This method would have to be modified for our purposes due to the possible differences in the constants C and G. One would have to do a force balance problem to calculate what the theoretical angles should be. This process could be automated and compared to physical experimental data.

Further, one is left asking if a five-to-ten second interval is good or bad amount of time for three drones to locate and converge on the operator, which likely will depend on the operator interaction necessary for a given mission or application. Spears does not address the time required to form the lattice into consideration as a performance metric. However, for our purposes, due to practical considerations such as battery life and a dynamically moving operator, convergence time to get into position is important. As such, future metrics need to take into account both time and lattice quality. Additional characterizations of swarm performance are likely to yield new research directions as well, as swarm algorithms are developed to address these objectives.

5.2.2 Additional Experiments

There are many follow-on experiments that would illuminate additional insights. One near-term experiment to be conducted should include one in which the coefficients associated

with drone-to-drone force and drone-to-operator force, G and C from Equation 3.4 and Equation 3.6, respectively, are varied incrementally to find the set of coefficients that minimize collisions and time required to find and create a formation around the operator. Incrementally changing the desired separation distance should also be explored. Another potential experiment that warrants data collection and analysis is an experiment designed to further test the way point attraction that is already built into the AP algorithm. Data should be collected both with an operator present and without an operator present. Additional functionality should be added that allows the drones to follow way points and avoid collision with the operator without attraction to the operator like under normal operation.

5.2.3 Continued Hand-held Controller and GUI Integration

The Wiimote is a good basic hand-held controller for basic drone group control. However, for more sophisticated interactions, its functionality should be extended to include a critical assessment of usability, i.e., what are the most useful button mappings. Specifically, further enhancements are required to the code that allows the user to take manual control of drones one at a time. Such modifications would reduce the amount of time required to set up trials and could lead to more interesting experiments where some drones are acting autonomously and some are under manual control. Another extension includes improving the GUI to have additional functionality such as point-and-click manual drone control, way point setting, and control over the constants used to calculate virtual force magnitudes. RVIZ is a good candidate for point-and-click manual control and way point setting. There exist built-in software libraries that allow for interactive markers in RVIZ, such that, e.g., a mouse click on a RVIZ screen could be translated into a command to move to a new position.

5.2.4 Altitude Control Issue Resolution

As previously discussed when multiple AR.Drones are in close proximity, there are frequent, sometime large (> 4 m), altitude excursions. We hypothesize that these excursions are due to pinger interference from the ultrasonic range sensing altimeter. Further experiments need to be conducted to verify this hypothesis and characterize this interference. If this is indeed the source of the problem, we need to examine if there is a software solution. This could require attempting modifications to the AR.Drone 1.0 firmware and altering the inner-loop altitude control feedback loop. Ideally, for indoor flight, altitude control feedback could come solely from Vicon, but would require software setting modifications to

enable circumventing the built-in altimeter.

5.2.5 Integration of AR.Drone 2.0

We collected all of our data using the AR Drone 1.0. Additional experiments should utilize the Advanced Robotic Systems Engineering Laboratory (ARSENL) AR Drone 2.0 fleet. Switching platforms may help with inner-loop altitude control issues and would allow longer flight times. The 2.0 Power Edition drones come with 1600 mAh batteries and can operate for approximately 18 minutes [35]. Longer flight times would greatly decrease the amount of time required to run repeated experiments and could enable the performance of more complicated mission based testing.

A laptop running Ubuntu 12.04 LTS with an Intel Core i5 processor and 1.9 GB of RAM was the main computer used for drone control and data logging. Upgrading to a more powerful machine could enable running experiments with more than three drones. Using several computers to comprise the GCS could also increase our ability to fly more drones simultaneously by splitting up processing tasks between the machines. We could connect these machines via a ROS network [47]. A ROS network allows multiple machines to share access to the same ROS topics. For example, one computer would be responsible for ingesting Vicon data and doing `tf` reference frame transformations. A second computer would use these transformations to calculate and transmit velocity commands to the drones.

5.2.6 Exception Handling

As currently coded, the AP node is not very robust. There is little or no logic for handling situations, for example, where an agent or operator leaves the Vicon coverage area. When this happens, the program currently fails and the drones continue to execute the last received command, which is usually not desirable. Of general applicability, there exists a need to develop code to command the drones back into the arena when they cross the boundary of the Vicon coverage area. One possible way to accomplish this is through dead reckoning. Dead reckoning could be used to estimate the drones position and orientation based on its received velocity commands and last good Vicon position and orientation measurement. Using the dead reckoned position, an algorithm could send new velocity commands to the agent to bring it back into Vicon coverage. Dead reckoned position would also update the AP node and could be used for calculating forces for other agents.

5.3 Future Work

5.3.1 Outdoor Flight

Currently, the Federal Aviation Administration (FAA), for civilian aircraft, and Naval Air Systems Command (NAVAIR), for Navy-owned platforms, place restrictions on UAV testing for research and development purposes. As of the publication of this thesis, the AR.Drone is not cleared for outdoor flight testing by Navy institutions. Assuming the AR.Drone does eventually receive an interim flight clearance, the next step in our research is to take the drones outside of the Vicon space and run the same experiments using GPS in a controlled outdoor environment. Parrot produces a GPS receiver for the AR.Drone 2.0 (see Figure 5.1). Note, however, that GPS is not yet implemented in the `ardrone_autonomy` ROS package. Future outdoor research would necessitate adapting this package to process and transmit GPS information via the WiFi connection. Taking the experiments outdoors would afford us the space to test the AP framework using more agents.

Figure 5.1: The Parrot Flight Recorder for the AR.Drone 2.0 has an on-board GPS, from [48].

5.3.2 Adding More Agents

Our research usually involves three or fewer AR.Drones with one human operator. Exceeding three agents with one controlling station appeared to result in too much latency for a satisfactory level of control. It is unclear if the latency is due to network bandwidth, controlling station processing limitations, `tf` algorithm limitations, or the Vicon motion capturing system. Using a controlling station with more processing power in conjunction with running a network packet monitoring program such as Wireshark could help identify

the bottleneck. As discussed in Section 3.3.2, sonar pinger interference is another major limiting factor in piloting multiple AR.Drones in a small arena. It may be necessary to modify the drone's firmware to remove pinger data from the altitude control algorithms.

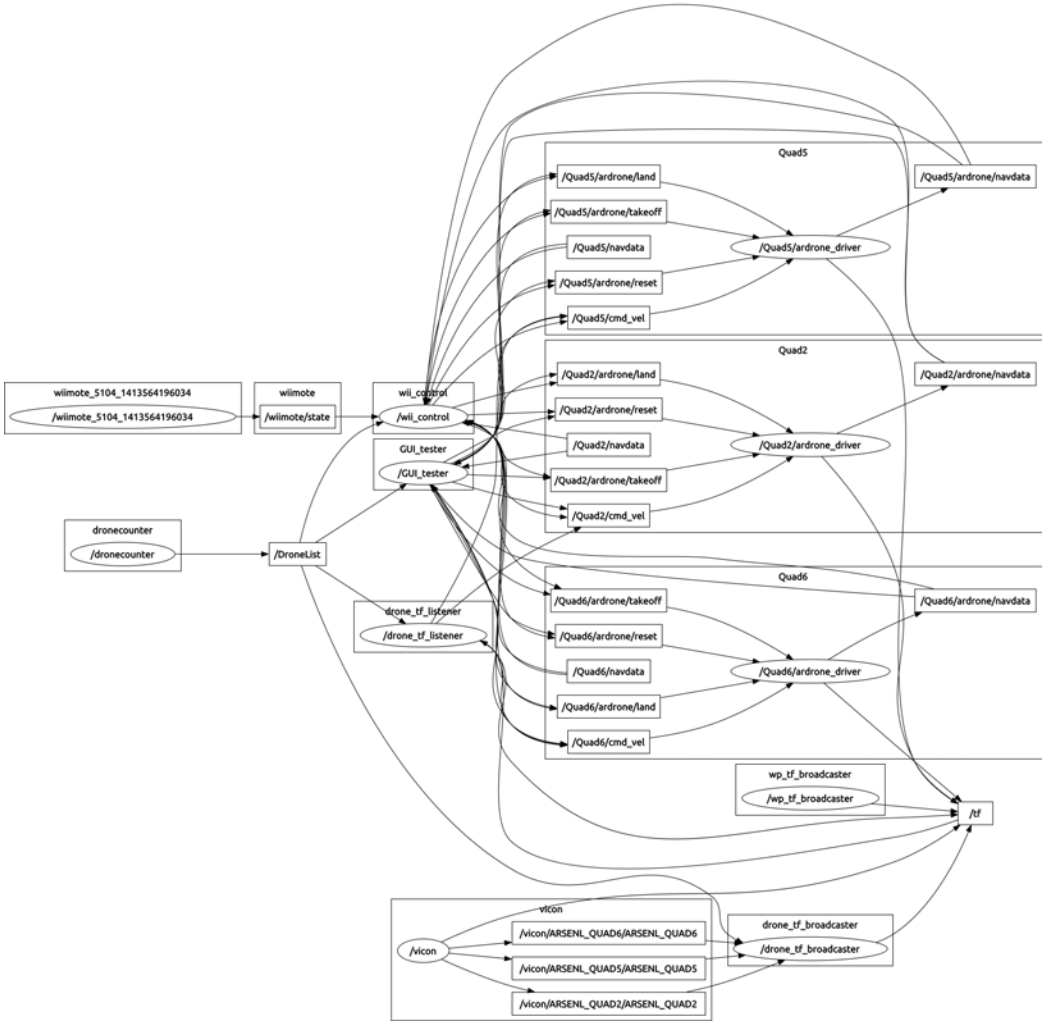
5.3.3 Tag Detection and Computer Vision

The AR.Drone 1.0 is equipped with two cameras. Using its own onboard processing power, the drone is able to detect the presence and relative position of special colored tags. The raw drone video feed can also be analyzed using tools like OpenCV to perform tasks such as localization, collision prevention, and threat detection. Future research should leverage the existing hardware on the drones and incorporate open source programs such as OpenCV. Incorporating tag detection and computer vision will help enable mission-based testing.

5.3.4 Mission-Based Testing

Part of the motivation for this project was to show that a group of drones could be useful in the field for following a group of operators and alerting them to threats. We need to design experiments that allow us to work up to this goal. Such experiments will leverage the drone's camera sensors and will need to be conducted outdoors due to space constraints in the existing NPS Vicon space. The computer or computers that comprise the GCS should be further designed so that an operator could carry them in a backpack.

APPENDIX A:
ROS Software Architecture



ROS node connection graph generated using `rqt_graph` [49]

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] J. Gertler, "U.S. unmanned aerial systems," Congressional Research Service, Washington, DC, CRS Report No. R42136, Jan. 2012. [Online]. Available: http://www.fas.org/sgp/crs/natsec/R42136.pdf

[2] E. Bonabeau and G. Théraulaz, "Swarm smarts," *Scientific American*, vol. 282, no. 3, pp. 54–61, 2000.

[3] T. De Wolf and T. Holvoet, "Emergence and self-organisation: a statement of similarities and differences," *Engineering Self-Organising Systems*, vol. 3464, pp. 1–15, 2004.

[4] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm robotics*. Springer, 2005, pp. 10–20.

[5] A. Martinoli, K. Easton, and W. Agassounon, "Modeling swarm robotic systems: A case study in collaborative distributed manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 415–436, 2004.

[6] H. Hamann and H. Wörn, "A framework of space–time continuous models for algorithm design in swarm robotics," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 209–239, 2008.

[7] F. Schweitzer, *Brownian agents and active particles: collective dynamics in the natural and social sciences*. Springer, 2007.

[8] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference, 1984*. IEEE, 1984, pp. 304–313.

[9] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Robotics and Automation, Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1398–1404.

[10] "Cooperative Payload Transport by Robot Collectives." [Online]. Available: http://www.eng.buffalo.edu/mechatronics/research/mobilemanipulator/index.html

[11] J. Canny and M. Lin, "An opportunistic global path planner," *Algorithmica*, 1993. [Online]. Available: http://link.springer.com/article/10.1007/BF01891836

[12] L. Barnes, M.-A. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Control & Automation, 2007. MED'07. Mediterranean Conference on*. IEEE, 2007, pp. 1–8.

[13] A. Tews and G. F. Wyeth, "Multi-robot coordination in the robot soccer environment," in *Proceedings of the Australian Conference on Robotics and Automation (ACRA'99), March*. Citeseer, 1999, pp. 90–95.

[14] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, "Graph laplacian potential and lyapunov functions for multi-agent systems," in *Cooperative Control of Multi-Agent Systems*. Springer, 2014, pp. 221–234.

[15] W. Spears, D. Spears, J. Hamann, and R. Heil, "Distributed, physics-based control of swarms of vehicles," *Autonomous Robots*, pp. 137–162, 2004. [Online]. Available: http://link.springer.com/article/10.1023/B:AURO.0000033970.96785.f2

[16] T. B. Apker and M. A. Potter, "Robotic swarms as solids, liquids and gasses." in *AAAI Fall Symposium: Human Control of Bioinspired Swarms*, 2012.

[17] T. B. Apker, J. Lennon, M. A. Potter, and D. Sofge, "Past point models: Physicomimetics on nonholonomic vehicles," *AIAA INFOTECH@ Aerospace*, 2011.

[18] T. B. Apker and M. A. Potter, "An artificial physics approach to plume detection with fixed wing uavs," in *AIAA Guidance, Navigation, and Control Conference*, 2011, pp. 08–11.

[19] E. Martinson and T. Apker, "Monitoring the auditory scene with self-organizing coherent microphone arrays."

[20] T. B. Apker and M. A. Potter, "Physicomimetic motion control of physically constrained agents," in *Physicomimetics*. Springer, 2012, pp. 413–437.

[21] S. Bashyal and G. K. Venayagamoorthy, "Human swarm interaction for radiation source search and localization," *2008 IEEE Swarm Intelligence Symposium*, pp. 1–8, Sep. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4668287

[22] L. Alboul, J. Saez-Pons, and J. Penders, "Mixed human-robot team navigation in the GUARDIANS project," in *2008 IEEE International Workshop on Safety, Security and Rescue Robotics*. IEEE, Oct. 2008, pp. 95–101. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4745884

[23] S. Bayraktar, G. E. Fainekos, and G. J. Pappas, "Experimental cooperative control of fixed-wing unmanned aerial vehicles," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4. IEEE, 2004, pp. 4292–4298.

[24] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for uav teams," in *AIAA 3rd unmanned unlimited technical conference, workshop and exhibit*, 2004.

[25] J. S. Jang and C. Tomlin, "Design and implementation of a low cost, hierarchical and modular avionics architecture for the dragonfly uavs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[26] D. T. Cole, S. Sukkarieh, A. H. Göktogan, H. Stone, and R. Hardwick-Jones, "The development of a real-time modular architecture for the control of uav teams," in *Field and Service Robotics*. Springer, 2006, pp. 465–476.

[27] P. Almeida, R. Bencatel, G. Gonçalves, and J. Sousa, "Multi-uav integration for coordinated missions," *Encontro Científico de Robótica, Guimarães, April*, 2006.

[28] T. Chung, "50 vs. 50 by 2015: Swarm vs. swarm uav live-fly competition at the naval postgraduate school," in *2013 Proceedings of AUSVI*.

[29] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," in *Southeastcon, 2012 Proceedings of IEEE*. IEEE, 2012, pp. 1–6.

[30] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[31] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[32] B. Remes, D. Hensen, F. van Tienen, C. De Wagter, E. van der Horst, and G. de Croon, "Paparazzi: how to make a swarm of parrot ar drones fly autonomously based on gps." in *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, 2013, pp. 17–20.

[33] M. S. Alvissalim, B. Zaman, Z. Hafizh, M. Ma'sum, G. Jati, W. Jatmiko, and P. Mursanto, "Swarm quadrotor robots for telecommunication network coverage area expansion in disaster area," in *SICE Annual Conference (SICE), 2012 Proceedings of*. IEEE, 2012, pp. 2256–2261.

[34] R. Purta, M. Dobski, A. Jaworski, and G. Madey, "A testbed for investigating the uav swarm command and control problem using dddas," *Procedia Computer Science*, vol. 18, pp. 2018–2027, 2013.

[35] S. Piskorski, N. Brulez, P. Eline, and F. D'Haeyer, "AR. Drone Developer Guide," 2012.

[36] "AR Drone by Parrot Archives - Drone Flyers." [Online]. Available: http://www.droneflyers.com/category/ar_drone/

[37] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[38] M. Hamer. (2012). *AutonomyLab/ardrone_autonomy · GitHub*. [Online]. Available: https://github.com/AutonomyLab/ardrone_autonomy

[39] "rviz - ROS Wiki." [Online]. Available: http://wiki.ros.org/rviz

[40] "Vicon | Homepage." [Online]. Available: http://www.vicon.com/

[41] R. B. Davis, "Applying cooperative localization to swarm uavs using an extended kalman filter," master's thesis, Naval Postgraduate School, Monterey, California, 2014.

[42] "vicon_bridge - ROS Wiki." [Online]. Available: http://wiki.ros.org/vicon_bridge

[43] T. Foote, "tf: The transform library," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, Apr. 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6556373

[44] "Hacked 10 Bits: Multiple AR.Drones from a single computer using the ardrone_autonomy ROS package." [Online]. Available: http://hacked10bits.blogspot.com/2014/02/multiple-ardrones-from-single-computer.html

[45] "wiimote - ROS Wiki." [Online]. Available: http://wiki.ros.org/wiimote

[46] "matplotlib: python plotting — Matplotlib 1.4.2 documentation." [Online]. Available: http://matplotlib.org/

[47] "ROS/NetworkSetup - ROS Wiki." [Online]. Available: http://wiki.ros.org/ROS/NetworkSetup

[48] "AR.Drone 2.0. Parrot new wi-fi quadricopter- Civil Drone with on-board GPS - Flight Recorder ." [Online]. Available: http://ardrone2.parrot.com/apps/flight-recorder/

[49] "rqt_graph - ROS Wiki." [Online]. Available: http://wiki.ros.org/rqt_graph

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California